Enhancing Network Intrusion Detection through Host Clustering

Master's thesis

UNIVERSITY OF TWENTE.

W.J.B. Beukema

July 2016

W.J.B. Beukema: *Enhancing Network Intrusion Detection through Host Clustering,* July 2016

SUPERVISORS:

Prof. dr. ir. Aiko Pras
Dr. Anna Sperotto
Luuk Hendriks MSc

Thomas Attema MSc (TNO)
Ir. Harm Schotanus (TNO)

Abstract

The state-of-the-art in intrusion detection mainly relies on signature-based techniques. Although signature-based detection is an efficient way of protecting against known threats, it will not protect against new, advanced intrusions such as Advanced Persistent Threats (APTs). Moreover, many intrusion detection systems only monitor the network traffic crossing the external border of a network, ignoring the internal network traffic. This research proposes a new approach towards detecting advanced attacks, by focusing on internal network traffic and by using anomaly-based detection. The performance of the anomaly detection is enhanced by using clustering techniques.

Internal network traffic is an undervalued source of information for recognising APT-style attacks. Whereas most systems focus on the external border of the network, we show that APT-style campaigns often involve internal network activity and that certain changes in internal network behaviour are a strong indicator for intrusions. To this end, a framework that shows the relation between attack characteristics and the impact on internal network traffic patterns is presented.

To reduce false positive rates and limit the burden of data processing, we propose an additional step in model-based anomaly detection involving host clustering. Through host clustering, individual hosts are grouped together on the basis of their behaviour on the internal network. We argue that a behavioural model for each cluster, compared to a model for each host or a single model for all hosts, performs better in terms of detecting potentially malicious behaviour. We show that by applying this concept to internal network traffic, the detection performance increases for both identifying malicious flows and identifying malicious hosts.

Contents

List of Tables, Figures and Listings

List of Acronyms and Abbreviations

ACC	Accuracy	67, 68
APT	Advanced Persistent Threat	4, 5, 7–11, 17, 18, 21–26, 51, 53, 54, 59, 66, 71, 73, 75
ARI	Adjusted Rand Index	36, 38, 40, 43, 44, 79
C&C	Command and Control	8, 18–24, 52
DHCP	Dynamic Host Configuration Protocol	50, 75
DNS	Domain Name System	12, 49, 50, 78
FPR	False Negative Rate	67, 68
IDPS	Intrusion Detection & Prevention System	10
IDS	Intrusion Detection System	10–15, 17, 59
IETF	Internet Engineering Task Force	49
IOC	Indicator of Compromise	13
IPFIX	IP Flow Information Export	49, 52, 53
IPS	Intrusion Prevention System	10
IPv4	Internet Protocol version 4	12, 53, 74, 75
IPv6	Internet Protocol version 6	12, 53, 74, 75
MCMC	Markov chain Monte Carlo	35
P2P	Peer-to-Peer	19
PCAP	Packet Capture	49, 50, 77, 78
RBF	Radial basis function	31, 55–57
ROC	Receiver Operating Characteristic	65–68, 79
SBM	Stochastic Blockmodel (clustering)	28, 34, 35, 38, 42, 64, 67–69, 79
SVM	Support Vector Machine	55–57, 69, 79
TPR	True Positive Rate	67, 68
UNB ISCX	University of New Brunswick, Information Security Centre of Excellence	39, 43, 61–64, 66–68

1 Introduction

In this chapter, an introduction is given to the problem behind the research of this thesis. The problem statement is discussed (section 1.1), followed by the introduction of the research questions (section 1.2). Furthermore, the proposed research methods are discussed in section 1.3. Finally, an overview of the structure of this thesis is provided in section 1.4.

1.1 Problem statement

1.1.1 Background

In the last decades, cyber-attacks have had a significant impact on the security of businesses and organisations. In 2014, the cost of cyber-attacks on world-scale was estimated to range from € 300 billion to € 600 billion annually [46]. Especially advanced, targeted cyber-attacks are a serious threat to businesses and organisations worldwide. As targeted attacks are getting more sophisticated every year, targeted attacks are amongst the biggest security threats to businesses and organisations [66].

A means to detect this kind of attack are intrusion detection systems. Even though there are various methods to detect intrusions, these advanced monitoring tools are dominantly misuse-based or anomaly-based. The former relies on signatures, which are created after human analysis of discovered threats. The latter relies on analysing a system's behaviour in an automated way, which makes it possible to detect unknown attacks.

Currently, most commercial intrusion detection systems are misuse-based [26]. An inevitable result of this approach is that detecting undiscovered threats is rather difficult, as there are no signatures available for these threats beforehand. Especially advanced, targeted threats, such as Advanced Persistent Threats, are hardly detectable using this approach, as they often use zero-day vulnerabilities.

As threats are likely to become stronger and more advanced in the long term, anomaly-based detection can provide a more flexible approach in keeping up with these threats compared to misuse-based detection.

1.1.2 The problem

The problem is that in order to detect advanced network intrusions such as Advanced Persistent Threats, a misuse-based defence is insufficient. For this reason, anomaly-based methods are necessary. However, the current anomaly-based detection methods used in this domain have severe shortcomings.

In some traditional network-based anomaly detection approaches, a model is created for each individual host in a network. Based on these models, the intrusion detection system analyses all network traffic for each host and tries to detect abnormalities. However, this approach suffers from several drawbacks. For instance, individual host models often result in high false positive rates, caused by the fact that minor changes in the host's behaviour may be considered to be anomalous. Additionally, this method does not scale well to large networks, making this approach impractical for deployment in larger networks.

Another approach is creating a single model for a network as a whole. Although this solves scalability issues, the pitfall of this approach is that by taking too many hosts together, important details may get lost. Consider a host that usually behaves as a workstation, suddenly starts behaving like a server. Although this would be certainly an anomaly, the behaviour the host in question shows in the new situation itself is not anomalous, as there are other hosts in the network showing the same behaviour. Therefore, it is unlikely to be flagged as anomalous using this approach.

Given the fact that anomaly-based detection is able to identify threats that misuse-based attacks cannot identify, research in the field of anomaly detection is of vital importance to keep up with the progressing cybersecurity landscape. Improvements to existing anomaly-based detection approaches, in particular with respect to detection rates, are necessary to achieve this.

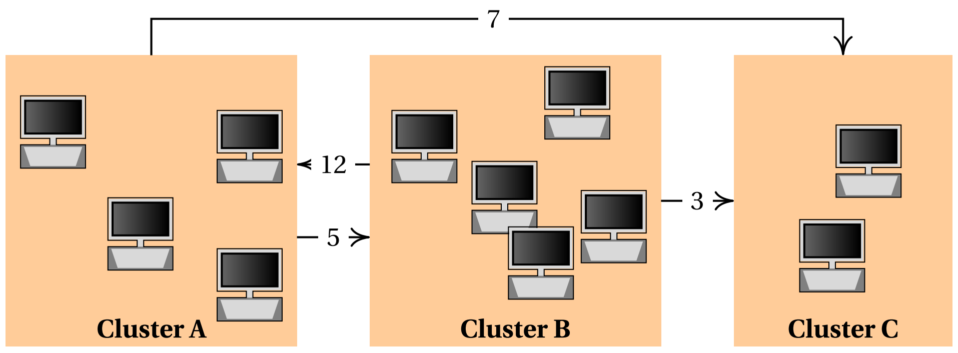
In other words, the problem is that despite currently being the best option to identify undiscovered threats, anomaly-based intrusion detection suffers from some serious limitations which may make its use impractical. The fundamental question therefore is how anomaly-based intrusion detection can be improved such that it is feasible to use it in practice.

1.1.3 Proposed solution

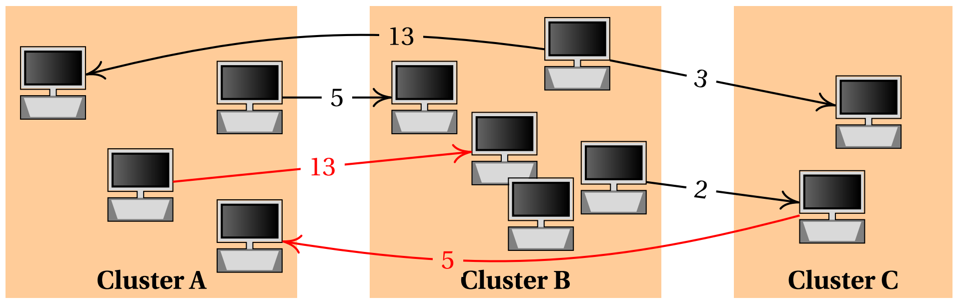
For this research, in trying to find which method provides the best way to find anomalies, it is investigated how host clustering can be used to enhance the performance of anomaly-based intrusion detection. To this end, we propose an enhancement to existing anomaly-based intrusion detection methods that involves clustering host entities acting on the internal network. When analysing a network of hosts, we propose grouping the entities into clusters, based on similarities between the entities. For each cluster, a model will be created that describes how entities within the cluster are expected to behave. Based on these models, it is examined how anomalous behaviour can be detected. To illustrate this concept, a simplified example is given in figure 1.1.

This concept of cluster-based modelling is applied to internal network traffic, i.e. network traffic between hosts on the local network that does not involve the (external) Internet. We argue in this work that internal network traffic is a valuable source of information to detect advanced intrusions and that our approach works well with internal network traffic data.

Being part of a larger network, hosts often share certain characteristics with other hosts. One could think of the assigned role a host has: a host could be a printer, a web server, a workstation, etc. Another example would be the connection partners host have: hosts often mainly communicate with a smaller subsection of the hosts within the network. Thirdly, hosts can be similar in the way they communicate on the network (level of network activity, number of bytes sent, etc.). With such characteristics in mind, it may be possible to consider a group of hosts as being one cluster that behaves in a certain way. Part of this research is to explore on what features a group of hosts should be clustered in order to get the best performance for anomaly detection.



(a) Sample model for a network consisting of three clusters. The edges indicate traffic flows between hosts in the clusters, the weight is a measure for the average intensity of the traffic. The hosts in each cluster are expected to adhere to their cluster's behaviour.



(b) Instance of interactions within the sample network, compared to the above model. Black interactions indicate behaviour that is (close to) the behaviour as described in the above model, while red interactions are anomalous. For simplicity, inner-cluster flows have been omitted.

Figure 1.1: Basic example of identifying anomalous behaviour on cluster-level internal network communication patterns. Note that the proposed anomaly detection system takes many more characteristics into account.

Model-based detection is not new in the domain of intrusion detection, as was discussed in the previous section. However, with our solution, an approach in between the existing per-host modelling and a single model is taken. It takes advantage of considering multiple hosts at one time, while trying to distinguish between different classes of hosts. To the knowledge of the author, no similar research has been conducted before.

We argue that the proposed approach overcomes both issues of high false positive rates and scalability through the clustering aspect. Grouping similar hosts is likely to well-represent the 'average' behaviour of these hosts, making small-changes in behaviour less anomalous. Also, because the number of clusters is strictly smaller than the number of hosts, fewer models have to be stored and therefore this method scales better to larger networks.

1.1.4 Contribution

The proposed approach allows organisations to improve their intrusion detection mechanisms. The novelty of this study is twofold: first of all, the presented approach is a new way to detect anomalous behaviour which performs better than existing approaches. Secondly, the presented approach is expected to scale better to large networks compared to existing approaches, resulting in less overhead.

More generally, the proposed technique of applying clustering can be applied in other domains than just internal network traffic. Even though this research focuses on internal network traffic, the proposed method may be applied to solutions such as external network traffic analysis, user behaviour analysis and host behaviour analysis. In this sense, a more general contribution of this research is providing a new way of enhancing anomaly detection when dealing with vast amounts of data.

1.2 Research questions

Based on the problem statement in the previous section, we derive the following main research question:

How well does internal network traffic anomaly detection based on host clustering perform in detecting network attacks such as APT*s?*

The first part, 'internal network traffic anomaly detection based on host clustering', refers to the methodology that we investigated. As part of this, methods have been developed that group network hosts into clusters, models 'normal' behaviour of hosts within each cluster, and subsequently detects abnormalities based on this modelled behaviour.

The performance of this approach is measured in terms of effectiveness (the expected false positive/false negative rate) and efficiency (the overhead/computational complexity of the algorithm). To rank this performance, both performance factors are compared with similar methods.

The focus of the developed methods are, as the last part of the research questions suggests, on detecting anomalies caused by advanced network attacks such as APTs. As has been mentioned before, these attacks are hard to detect using traditional mechanisms and are therefore an interesting topic for research.

From this, the following sub research questions are derived:

1. How do targeted network intrusions impact internal network traffic patterns?
2. How can hosts within a network be divided into clusters of hosts with similar behaviour?
3. How can models, representing normal behaviour, be derived from network host clusters?
4. How can the behaviour of clusters be used to identify anomalous behaviour?
5. How well does the suggested approach perform in terms of detection rates and overhead?

1.3 Research methods

To be able to answer the research questions, different research methods are used. The first subquestion concerns an analysis of the effect of APTs on internal network traffic patterns, and will be answered by a literature analysis of known APT campaigns. The second research subquestion comprises a literature review of existing clustering methods and an evaluation of those for the domain of this research. The third subquestion involves a literature analysis of model-based intrusion detection, accompanied by an applied design for this domain, based on the previous subquestions. For the fourth subquestion, the preceding subquestions are used to design an algorithm that uses cluster-models for anomaly detection on internal network traffic.

Finally, for the fifth subquestion the developed algorithm is evaluated in order to be able to make statements about its performance.

An overview of the proposed methods per research question is provided in table 1.2.

Table 1.2: Research methods per subquestion, as outlined in section 1.2.

Research subquestion	**Research method**	**Chapter**
How do targeted network intrusions impact internal network traffic patterns?	Literature analysis	Chapter 3
How can hosts within a network be divided into clusters of hosts with similar behaviour?	Literature analysis and evaluation	Chapter 4
How can models, representing normal behaviour, be derived from network host clusters?	Literature analysis and design	Section 5.1-5.2
How can the behaviour of clusters be used to identify anomalous behaviour?	Design	Section 5.3
How well does the suggested approach perform in terms of detection rates and overhead?	Evaluation	Chapter 6

1.4 Structure of this thesis

The next chapters will address the different research subquestions. Chapter 2 discusses the state-of-the-art in this field of research by introducing the concept of Advanced Persistent Threats and providing a literature review on existing intrusion detection techniques. Chapter 3 systematically reviews a number of APT campaigns and evaluates their impact on internal network traffic, in order to understand their relation. Chapter 4 introduces clustering as a concept, discusses a number of clustering methods and evaluates their applicability to this domain. In chapter 5, the subjects of APTs, intrusion detection and clustering are brought together as is investigated how host-clustering can be used to detect APTs. This chapter also features the proposed solution. An evaluation of this proposed solution is evaluated in chapter 6, preceded by an evaluation of available data sets for this purpose. Chapter 7 addresses the conclusions of the research questions based on the results of the previous chapters. Finally, limitations of this research and other points of discussion are presented in chapter 8.

2 State-of-the-art

In order to provide some background information for the context of this research, this chapter will discuss the current state of research regarding Advanced Persistent Threats (section 2.1) and give a literature overview of intrusion detection system techniques (section 2.2).

2.1 Advanced Persistent Threats

2.1.1 Definition

Advanced Persistent Threat (APT) is a type of cyber attack, which is usually managed by well-skilled and well-funded attackers, often targeting specific organisations or sectors. Goals APT campaigns may have include stealing information, theft of assets, and damaging the target. Whereas APTs used to be directed at political and military targets, there has been a shift towards enterprise targets [24].

APTs are characterised by the following:

Advanced The adversary is highly skilled, and has significant resources. In contrast to traditional malware, APTs are targeted: they are specifically designed for the target and involve manual actions from the attacker's side.

Persistent The adversary is able to infiltrate in to a network, where it remains hidden for a long time (either passively or actively). Using a Command and Control channel, the attackers are able to continuously monitor, extract data from and send commands to the target.

Threat Traditionally, for APTs the adversary's goal is to obtain sensitive information, such as intellectual property, trade information or political information. However, more recent attacks show attackers may have different goals, such as damaging or destroying the target, manipulating data of the target, etc.

The unusual stealth, advanced skill set of the attackers and the vast resources available to them result in a powerful type of attack with a remarkably high success rate [7].

The term APT was coined in 2006 by analysts of the United States Air Force to describe complex cyber-attacks against specific targets over a longer period of time [7]. Roughly a decade later on, the term 'Advanced Persistent Threat' is still used in the cybersecurity domain to describe this kind of threat. However, there exists controversy on whether it is the right term for the type of threat it tries to describe. Some prefer the term *targeted attack*, as the 'targeted' element is a rather important feature that is not part of the APT acronym. Strictly speaking however, APTs form a subset of targeted attacks, containing the most advanced types of targeted attacks, as is shown in figure 2.1. For instance, cyber espionage or spear phishing in itself can be seen as targeted attacks, while they are not necessarily APTs.

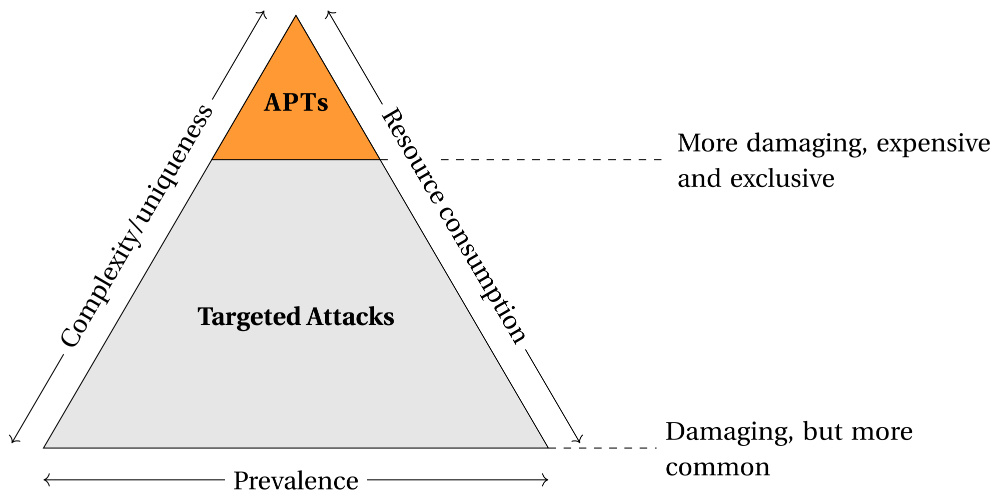


Figure 2.1: The relationship between APTs *and targeted attacks.*

2.1.2 Life cycle

There has been a lot of research in the typical behaviour of APTs and the various stages an attack can be in. For instance, Hutchins et al. [35] present a life cycle for advanced intrusions, accompanied with a Courses of Action Matrix, providing a framework for risk managers in detecting and mitigating such attacks. Brewer [7] provide a general framework for describing the phases of intrusion in case of a targeted computer network attack, which are discussed below. The Dell SecureWorks APT life cycle [20], another often-used model, shows strong similarities with Brewer's framework. A visualisation of the Dell SecureWorks model is shown in figure 2.2.

The stages described by Brewer and their link to the Dell SecureWorks life cycle are:

Reconnaissance The adversary gathers as much information as possible on the target. This includes, but is not limited to, the crawling of websites for information, collecting email addresses, structuring social relationships, and information concerning specific technologies used by the target. This phase matches the first five stages of figure 2.2.

Compromising the target Based on the gathered information, the adversary will select one or more entry points to gain foothold in the network environment. For this, the adversary will create malware that will be used for intruding on the network. This could, for instance, be a malicious attachment (e.g. a PDF file), a malicious applet (e.g. using an Adobe Flash vulnerability) or a trojan. This may involve the use of one or more zero-day exploits. To compromise the network, often social-engineering techniques are used. One way would be to use *spear phishing*, i.e. luring a user (that has access to the target environment) to open a specially crafted website or attachment to infect the system. Alternatively, a user could be tricked into attaching a malicious USB device to a system connected to the network. This phase matches the *Deployment* and *Initial intrusion* stages of figure 2.2.

Increasing foothold As soon as the attacker has compromised a target machine, it will do its best to maintain access to the systems. For this cause, the adversary will infiltrate deeper into the network to infect more machines. This may be achieved by obtaining access credentials through guessing passwords. Alternatively, this may be achieved by carrying out new attacks with the objective to obtain access to another part of the system, or to get higher privileges. If the APT succeeds, it might maintain access, even if one or more

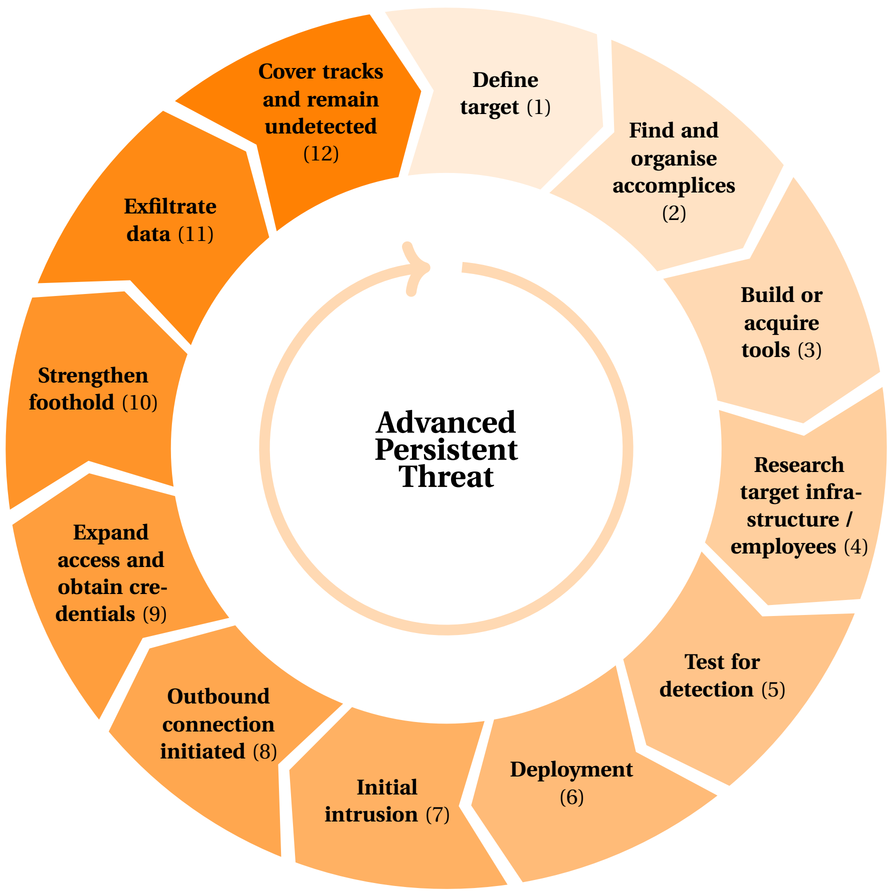

Advanced Persistent Threat
Define target (1)
Find and organise accomplices (2)
Build or acquire tools (3)
Research target infra-structure / employees (4)
Test for detection (5)
Deployment (6)
Initial intrusion (7)
Outbound connection initiated (8)
Expand access and obtain cre-dentials (9)
Strengthen foothold (10)
Exfiltrate data (11)
Cover tracks and remain undetected (12)

Figure 2.2: Overview of the SecureWorks APT *life cycle. Adapted from [20].*

infections are detected. Often, a command and control (C&C) channel is set up so that the adversary can remotely give instructions to the infected machines.

Lateral movement After establishing its foothold, the attacker will move around the network in order to identify the system or systems that are valuable to them. Often, the APT will try to move around to other systems using default credentials or by guessing passwords. Therefore, it may be less likely to be discovered at this stage. Note that unlike untargeted attacks, this stage is usually carried out manually. This and the previous phase match the *Outbound connection initiated*, *Expand access and obtain credentials* and *Strengthen foothold* stages of figure 2.2.

Carrying out attack If the attacker's goal is to obtain sensitive information, the attackers will try to move the target data outside the network once found. Again, to make this action difficult to detect, evasion techniques are applied, e.g. encrypting the target data before moving it around the network, or hiding a file using steganography in a legitimate looking file.

Alternatively, the attacker may manipulate data during this stage. When the adversary's goal is to destroy, damage or disrupt, the attack usually happens as soon as the attacker reached the right system(s). Depending on the goal, this stage might (in contrast to stealing information) immediately attract a lot of attention as a result of the attack.

This phase matches the *Exfiltrate data* stage of figure 2.2.

Hiding Often, after APTs have achieved their goal, they try to hide their tracks. As they have control over various systems, they may try to alter log files to make it appear that the APT was never there. This makes analysing APTs afterwards hard, as the available evidence may be tampered with. This phase matches the final stage of figure 2.2.

2.2 Intrusion detection techniques

From section 2.1, it follows that detecting APTs is harder than traditional malware, as a result of the sophistication of the attacks.

A class of solutions that intend to detect suspicious behaviour in a network environment are called *Intrusion Detection Systems* (IDSs). Usually, IDSs gather information from various types of system and or network sources. Subsequently, the information is analysed in order to identify possible intrusions on the system monitored [21]. Typically, IDSs only flag suspicious behaviour, and notify system administrators to take further actions. This is in contrast to *Intrusion Prevention Systems* (IPSs), in which case the system itself takes action to mitigate a possible attack, for instance by blocking suspicious connections. Due to the high similarity between IDSs and IPSs, they are sometimes called *Intrusion Detection and Prevention Systems* (IDPSs) in literature. In this analysis however, the focus will be on IDSs.

IDSs come in many different forms using different methods and algorithms. In the next subsections, a number of important characteristics are discussed. An overview of the taxonomy of IDSs can be found in figure 2.3.

2.2.1 Data source

The first factor on which IDSs differ, is the source of the data on which the IDS makes its decisions. IDSs are usually host-based or network-based, although hybrid options combining the two exist as well.

2.2.1.1 Host-based

Host-based IDSs work, as the name suggests, on single hosts to detect suspicious behaviour. The events and behaviour of a single host are monitored for suspicious activity. Usually, this entails monitoring log files, running processes, file access, file modification/deletion and system configuration changes. Also, this may include incoming and outgoing network traffic of the host [62].

Host-based intrusion detection is mostly installed on the host itself. Traditional detection and prevention systems, such as virus scanners and firewalls, could be seen as host-based intrusion detection and prevention systems.

The advantage of this approach is the level of detail available to the IDS: virtually everything the machine is doing can be monitored by the IDS. In contrast to network-based approaches[1], the contents of packets can always be analysed at application-layer level. A downside of a host-based approach is the overhead it causes on the resources of the machine. Even more disadvantageous is the limited scope of host-based IDSs: in a network of hosts communicating with each other, a host-based IDS would only look at individual hosts and not take the bigger picture into account. This could as a result reduce the effectiveness of the solution.

[1]See section 2.2.1.2.

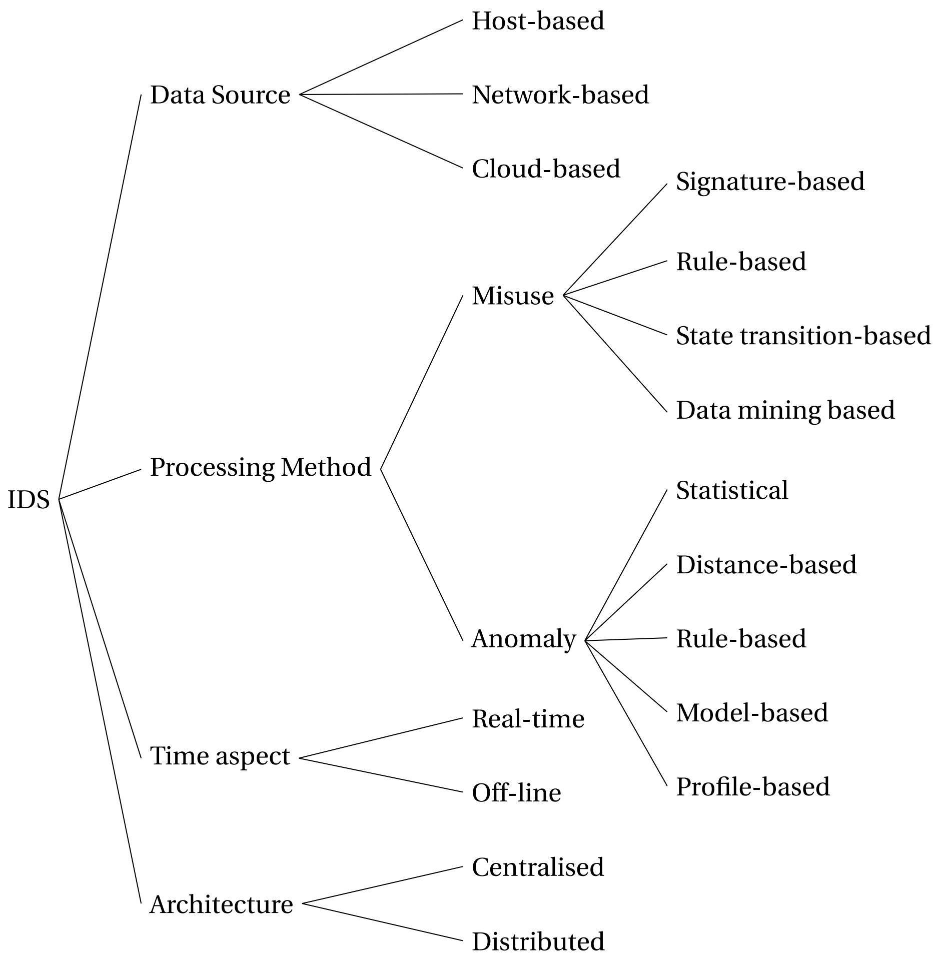


Figure 2.3: Taxonomy of the different characteristics of IDSs*, adapted from [61].*

2.2.1.2 Network-based

As a main characteristic of APTs is the strong network component, it may be beneficial to look from a network-level when trying to detect advanced network threats. Network-based detection approaches look at parts of a network to analyse the traffic and protocol data in order to detect intrusions.

Typically, a network-based IDS consists of multiple sensors, which monitor and analyse network activity on one or more network segments [62]. Generally, there are two ways in which sensors are deployed. One way is to force all network traffic to pass through the sensor, making it an *inline sensor*. This type of sensor is usually combined with a firewall, which uses the same approach. As a result, this approach makes it possible to filter malicious traffic. The downside of this approach is that it might create a bottle neck at the sensor, delaying the network traffic if the sensor cannot process the packets quickly enough. Another option would be to copy network traffic without actually making the actual traffic pass the detector, i.e. a *parallel sensor*. In this case, an alert may be given when a suspicious network flow is detected, upon which action can be undertaken by the user; the IDS itself cannot actively intervene.

Network-based intrusion detection can be applied to external network traffic (i.e. traffic between the internal network and the 'external' Internet), internal network traffic (i.e. traffic inside the internal network) or both. In practice, most existing solutions focus on external network traffic [3].

The data processed by this type of IDS is solely network data. For each network layer (e.g. OSI model layers [72]), an IDS could potentially extract information which it may use to identify threats, such as:

Data Link Layer This layer handles the frames exchanged between physical components of the network. The data link layer is rarely taken into account by network-based IDSs.

Network layer Responsible for addressing and routing network data, the network layer provides information such as source and destination IP addresses. Next to the IPV4 and IPV6 protocol running on this layer, the Internet Control Message Protocol (ICMP) is also used. Next to the IP address information, additional fields from the packet headers (such as Time to Live) may be useful in characterising network patterns.

Transport Layer This layer is responsible for transmitting data between hosts, and is therefore useful for network-based approaches in order to identify communication patterns. On this layer, TCP and UDP are mainly used. Each TCP segment and UDP datagram has a port number for both source and destination. Due to the connection-oriented nature of TCP, information about the session between source and destination can also be used in tracking communication patterns.

Application Layer Even a step further would be to inspect the contents of the application layer. Analysing data on this level is also called *Deep Packet inspection*, emphasising the depth of the analysis. Application-specific information from protocols such as Hypertext Transfer Protocol (HTTP), Domain Name System (DNS), and Simple Mail Transfer Protocol (SMTP) may be used to identify threats. Some protocols, such as SSL/TLS and SSH, make inspection at this level harder, as the actual payload carried by this layer is encrypted.

An advantage of network-based intrusion detection is that it offers extensive detection capabilities. Running intrusion detection from a network level perspective provides a view on the 'bigger picture', whereas host-based solutions usually only take individual hosts into account. Running sensors in a network will, if deployed correctly, not reduce the performance of the hosts. A network-based approach is also more flexible compared to host-based approached in the sense that it does not require new hosts to be configured in order to be monitored. Every device connected to the network will be automatically be monitored.

A drawback of this approach is that under high loads, the IDS may either not be able to process all traffic or will cause network traffic congestion. The former could apply to passive sensors, simply dropping packets, while the latter could apply to inline sensors. Hence, it is important that a network-based IDS has enough capacity to process items, possibly scaling with the network load. Another issue, which has already been addressed, is the possibility that encryption is applied at application-layer level, making it harder to carry out a full analysis on the network traffic. Especially for forensic-level analysis, this is a downside. Nevertheless, information gathered from the other levels can still be analysed, which is considered to provide sufficient information to identify intrusions [61].

2.2.1.3 Cloud-based

An emerging trend is the use of cloud-based IDSs [60]. Cloud-based IDSs can be deployed similarly to network-based IDSs, using sensors in the organisation's network. Instead of processing and analysing the data on-premise, the data is transferred to the cloud provider.

While the advantage of on-premises deployment of an IDS is the complete control over all the systems and data, the downside is the cost on hardware, software and maintenance. A cloud-based IDS can offer increased scalability, speed and support, as it is often provided by a specialised company on a large scale. An additional advantage is that the organisation does not have to keep up with the latest technologies and functionalities in order to ensure an acceptable level of security.

Even more interesting, cloud-based IDS providers often centrally correlate the data gathered at their clients' site with multiple intelligence sources at once [52]. This makes it also possible to use the data to detect new, undiscovered threats, with the additional advantage of being able to apply the intelligence directly to all customers. Thus, in terms of analysing and researching new attacks, cloud-based IDSs provide the most flexible mechanism. However, since the captured data has to leave the premises, some organisations fear privacy issues, making it the main barrier for deploying this type of IDS.

2.2.2 Processing method

Sabahi and Movaghar [61] distinguish between misuse methods and anomaly methods with regards to intrusion detection. Also, within those methods, a number of subtypes can be specified.

2.2.2.1 Misuse

Misuse detection is based on knowledge of known attacks and system vulnerabilities, often provided by human experts. Within this approach, it is usually necessary to have extensive knowledge on the common practises and to keep up-to-date with developments in order to provide security against intrusions [42].

Signature-based methods Within a signature-based approach, the IDS tries to find so-called signatures in the observed behaviour. The signatures are a predefined set of patterns of known intrusions, maintained in a database by the IDS. Therefore, only known attacks can be found; new attacks need to be updated over time.

Rule-based methods A rule-based system relies on a set of *if-then* rules to characterise computer attacks. Similarly to signature-based methods, this is often based on known attacks or known attack vectors.

State transition-based methods In a state transition system, the IDS maintains a final state machine in which each state corresponds to a state within the IDS, often based upon network protocol stacks or integrity and validity of files or running processes. When a flagged state is reached, the situation is reported as a possible intrusion.

Data mining based techniques In a data mining approach, the IDS bases its decisions on a database with past events, all labelled either 'normal' or 'intrusive'. A learning algorithms will train itself based on this database. Based on this information, the IDS will recognise known attacks or variations on it. The advantage of this approach is that models of misuse are created automatically and is better able to detect variations (in contrast to the approaches above).

A specialisation of misuse-based detection is *indicator of compromise* (IOC), an approach in which an IDS tries to find evidence for an intrusion. Typically, this is achieved through virus signatures, but also IP addresses related to known malware and hashes of malware files.

Details of IOCs can be relatively easily be shared and used for (manual) analysis or automated detection. In an effort to quickly exchange indicator of compromise data among systems, there have been initiatives to standardise a format describing the indicator(s), sharing the threat intelligence [17].

2.2.2.2 Anomaly

Anomaly-based intrusion detection tries to identify deviating patterns, comparing test data to a defined baseline of normal behaviour [42]. Compared to misuse detection, anomaly-based intrusion detection offers a more dynamic, flexible approach, as it may also detect unknown intrusions.

Statistical methods By measuring certain variables over time (e.g. traffic intensity, number of open connections, time a user is logged in for, keystrokes, etc.) statistics are applied to find deviations. This can for instance be achieved by keeping the average of the variables, and comparing how the new situation relates to these averages. If new values differ too much from the averages, it is marked as a possible intrusion. Alternatively, the IDS can derive a probability distribution for the variables and evaluate the probabilities for the new situation.

Distance-based methods Distance-based anomaly detection tries to detect anomalies by computing distances between multidimensional points representing several variables. Major approaches include nearest-neighbour, density-based and clustering. If a certain point, representing an analysed event, cannot be linked to existing clusters, it can be considered as abnormal behaviour. The main advantage of distance-based over statistical methods is that is more flexible in taking multiple variables into account.

Rule-based methods IDSs relying on rule-based anomaly detection have predefined knowledge of what is considered to be normal behaviour. When a host deviates from this pattern, it is identified as a possible intrusion. Because of the statical approach, this kind of anomaly detection will not automatically adapt to changing situations.

Model-based methods Similar to rule-based intrusion detection, in a model-based approach the normal behaviour of a monitored system is defined and formalised into a model. An anomaly is detected as deviation of the model.

Profile-based methods Profile-based intrusion detection has a profile of normal behaviour for each type of network traffic, hosts, program, etc. based on historic data. Profiles are usually build using data mining techniques or heuristic-based approaches. Again, when is deviated from this pattern, the IDS will mark this as a possible intrusion.

2.2.2.3 Comparison and *status quo*

An vital factor for the processing method is the accuracy, in other words, the balance between true and false positives and true and false negatives. IDSs can often be configured by setting parameters that represent the ‘sensitivity’ of the algorithms used: if set too high, the IDS system will face lots of false positives, whereas setting it too low will result in a high number of false negatives. Neither are desirable: the former will result in an unnecessary burden for those operating the IDS, the latter will result in a compromised system (going beyond the point of using an IDS). Thus, these rates determine how effective a solution is, as does it say something about the usability. It should be noted however that the environment also plays a role in this:

for instance, an organisation with a high security level is likely to accept a high false positive rate if this means it increases the true positive rate.

As mentioned before, misused-based approaches require manual input and do not account for new, undiscovered threats, in contrast to anomaly-based approaches. Anomaly-based approaches however usually have higher false positive rates compared to misuse-based [3], as unexpected but harmless behaviour may be seen as a potential threat.

In this day and age, there are various IDSs available, such as those provided by FireEye, Dell Secureworks, Fortinet and Trend Micro [60]. Despite the significant research in the field of anomaly detection, modern intrusion detection systems tend to focus on misuse techniques, with none or very little anomaly techniques. Mainstream solutions therefore rely on the intelligence provided by the provider, which in turn relies on manual analysis of known threats.

2.2.3 Other

Additionally, IDSs can differ in the moment at which the detection is applied. Usually, an IDS will detect intrusions real-time or near real-time. In this case, the IDS will work with the data as it comes in, goes through and goes out the network on-the-fly. An alarm will be raised as soon as a possible intrusion is detected, so that sufficient countermeasures can be taken if necessary. Alternatively, an offline IDS will analyse network data at a later point of time. This post-analysis is usually performed as a form of audit. Although offline IDSs reduce the overhead burden on a network, it is not considered to be sufficient to ensure high security [42].

Another distinguishing feature of IDSs is the architecture they use. Usually, IDSs have a centralised architecture. In this scenario, the analysis of data is performed on a fixed number of locations, independent of how many hosts are being monitored. Often, intrusions are detected that occur in a single monitored system. In a distributed IDS, the number of monitoring locations is proportional to the number of monitored hosts. In contrast to centralised IDSs, distributed IDSs can combine data from multiple network sites to effectively identify more advanced attacks [42]. This principle is used in cloud-based IDSs (see section 2.2.1.3).

3 APTs in internal networks

This chapter explores the impact of Advanced Persistent Threats (introduced in section 2.1) on internal network traffic. First, an introduction of the relevance of internal network data on intrusion detection is discussed (section 3.1), after which a number of APT campaigns are structurally analysed (section 3.2). Next, a framework is presented that describes the relationship between APTs and internal network traffic (section 3.3). Finally, conclusions on this chapter are drawn in section 3.4.

3.1 Internal network traffic

One of the characteristic features of an APT is the tendency to target network infrastructures instead of targeting individual hosts. When considering the APT life cycle (figure 2.2), especially the *Expand access and obtain credentials*, *Strengthen foothold* and *Exfiltrate data* phases are likely to involve internal network traffic.

What follows from the analysis in section 2.2, is that common intrusion detection mechanisms are misuse-based, and even those mechanisms that are anomaly-based are usually network-based and focus on data between the border of the internal network and the external network. Intrusions found using these approaches are likely to be in the *Initial intrusion*, *Outbound connection initiated* or *Exfiltrate data* phases.

If the existing intrusion mechanisms would have a 100% success rate at detecting APT-style attacks, the attack would be detected at an early stage, therefore not requiring any additional detection measures. However, since a perfect success rate is unlikely to be realised, one should consider how false-negatives can be detected at later stages. This is in line with the defence-in-depth principle.

With existing, common IDS technologies, networks are protected according to the eggshell principle [64]: strong on the outside, soft on the inside. In other words, the border between the internal network and the (external) Internet is well-monitored, while the activity of hosts on the internal network is hardly taken into account. This implies that once an attacker has gained foothold in the target environment, it can relatively easily move around the network in order to expand its access, making it more likely the attacker will be able achieve its goal.

As intrusion detection ought not to solely depend on intercepting APTs in the *Initial intrusion* and *Outbound connection initiated* phases, it is vital to develop methods to detect intrusions in the later stages as well. As these stages seem to have a strong component of internal network traffic, it is worthwhile to investigate this area further.

In the remainder of this chapter, it will be investigated whether and how existing Advanced

Persistent Threats use the internal network infrastructure as part of the threat and how internal network patterns change as a result.

3.2 Attack analysis

In order to gain insight into how APTs operate, this section will provide a literature review of APT campaigns.

When it comes to studying APT campaigns, it is often only possible to perform a black box analysis on the motivations behind the attack and the used methods. This is mainly due to the fact that for most APTs it is unclear whom are behind it, what their real motivations are and what their exact *modus operandi* is. The lack of this background information means that the analysis of APTs often comes down to contemplating what happened based on an interpretation of the available facts.

For this review, literature that (a) describes campaigns that are conform to the definition of APTs (as in section 2.1), (b) provides a sufficient level of detail to evaluate their impact on internal network traffic, and (c) is from a authoritative source, were selected.

As there is a very limited number of peer-reviewed literature available that describe the inner workings of APTs in detail, other literature is taken into account as well. Research institutes such as the Laboratory of Cryptography and System Security (CrySyS Lab, Budapest University of Technology and Economics) and cybersecurity companies such as Kaspersky and Symantec provide detailed threat reports on some campaigns in which systematically is assessed how the APT malware works and how the attackers operated throughout the campaigns.

Based on these requirements, the following four APT campaigns will be discussed: Stuxnet (section 3.2.1), Duqu (section 3.2.2), Flame (section 3.2.3), and Carbanak (section 3.2.4). Each campaign is introduced by a short overview, followed by an analysis of its behaviour per life cycle stage. Also, an overview of the APT's effect in internal network traffic is provided.

3.2.1 Stuxnet

Stuxnet [41] is a threat first discovered in July 2010. Analysis showed the target was an Iranian nuclear facility that enriched uranium. Stuxnet was designed to change the rotor speed of the centrifuges in such a way that it would destroy the centrifuges, and thus disturbing Iran's nuclear programme. It did so by infecting industrial control systems (ICSs), which control the hardware on which such facilities are ran (called PLCs).

An extensive report on the threat by Falliere et al. [23] describes in detail the different stages of the attack. In order to get to get into the facility, Stuxnet replicated itself through removable drives (such as USB drives) and local networks. This was achieved through exploiting several zero-day vulnerabilities in the Microsoft Windows operating system. When an infected host is connected to an ICS, Stuxnet secretly installs malware on the connected PLC and alters the control software on the infected host to disguise its presence. However, as Stuxnet only targets a very specific type of PLC, researchers were able to link Stuxnet to the Iranian nuclear facility [41]. Atypical about this APT is that it does not strongly rely on Command and Control (C&C) channels but instead works mainly offline, based on instructions generated before the malware was compiled. This was necessary due to the fact that the target was not connected to the Internet.

3.2.1.1 Attack phases

Reconnaissance The researchers of the Symantec report [23] concluded that the attackers must have obtained the schematics of the ICS as part of the reconnaissance phase, which enabled the attackers to adjust Stuxnet such that it would successfully destroy the centrifuges. It is highly likely that the attackers used a mirrored environment with the ICS hardware to test their code. Based on the analysis of Stuxnet's executables, it was found that Stuxnet targeted five different organisations.

Compromising the target It is highly likely that the initial intrusion took place by attaching an infected removable drive, supposedly at clients of the targeted organisations. The attackers managed to infect partners of the five target organisations, which in turn infected the target organisations through their removable drives.

Increasing foothold / lateral movement Given the environment Stuxnet intends to operate in (i.e. computers not having access to the external Internet or not networked at all), it propagates itself through LAN and removable drives to increase foothold.

Stuxnet contains many features that enabled it to spread itself through the network. It abused vulnerabilities in the Windows Print Spooler, SMB and WinCC, and copies/executes itself on remote computers through network shares. The latter entails trying to copy itself to all available network resources and execute on the remote share using either security credentials found on the local computer or domain, or through a WMI Explorer impersonation.

Performing the attack Because of Stuxnet's target environment, the code to sabotage the target systems is part of the Stuxnet executable itself. C&C channels are used to instruct infected machines that have access to the external Internet, while the target systems (which often do not have such access) cannot be controlled in this way. However, Stuxnet uses a peer-to-peer mechanism to update the executable on all machines of an infected network. An infected host that has access to the Internet could get instructed to download a new version of the executable and distribute it to the other hosts on the internal network. Through this, the attackers are still able to some extent command/control all infected machines, although not directly or machine-specific.

In brief, the P2P structure was set up as follows: every infected machine installs and starts an RPC server and client. Any other compromised host on the network can connect to the RPC server to query the installed version of Stuxnet on the remote host. If the remote version is newer, then the local computer will request a copy of the new version and will update itself. If the remote version is older, the local computer will prepare a copy and send it to the remote computer.

Covering tracks Stuxnet has a built-in, fixed kill date of June 24, 2012. From that day on, it will no longer infect machines through removable drives. Additionally, it tries to erase itself from infected machines on that date.

3.2.1.2 Network data behaviour

A few observations regarding the network aspects of Stuxnet:

- Stuxnet uses several methods to spread itself through the network, all of which may result in anomalous network behaviour if the performed action is uncommon. For instance, the

Windows Print Spooler vulnerability sends a WPS request to a host with a shared printer. If the infected host never exchanged data with the destination or never used WPS on the destination host before, the action is likely to be marked as anomalous.

- The peer-to-peer mechanism to propagate updated versions of Stuxnet through a local network can also result in anomalous behaviour, as Stuxnet does not take into account which hosts an infected host usually communicates with.
- Stuxnet may build a complete peer-to-peer network, meaning that a host may have multiple peers. As will become apparent in the next section, this is different from the peer-to-peer mechanism as used in Duqu, in which a peer-to-peer network seems to consist of one middle man that helps one or more (individual) hosts.

3.2.2 Duqu

Closely related to Stuxnet is Duqu [2]. Discovered in 2011, Duqu appears to target machines that may have interesting information with regards to attacking ICSs.

According to Bencsáth et al. [2], Duqu re-uses a lot of source code found in Stuxnet. However, unlike Stuxnet, Duqu only seems to capture system information (including key strokes, screenshots, browser history and system logs), whereas Stuxnet's main goal seemed to be destroying a certain type of ICS. Duqu would infect machines through a zero-day vulnerability in Microsoft Word. Duqu-infected machines could receive instructions from the attackers over a C&C channel. Communication over the channel took place over HTTP, obfuscated as JPEG images. After 30 to 36 days, Duqu would self-destruct itself in order to limit traceability.

3.2.2.1 Attack phases

Reconnaissance Even though Duqu does not seem to have a specific target, it follows from analysis that it is targeting intelligence data and assets from entities such as industrial infrastructure and system manufacturers. Possibly, the goal of this is to conduct a future attack against another party relying on these systems [23].

Compromising the target In order to infect a target, spear phishing was used. For this, a zero-day injected Microsoft Word document was used, that takes advantage of the way the Windows kernel used to handle embedded fonts. Through this, Duqu installed itself on the target machine.

Increasing foothold / lateral movement Duqu can be commanded to replicate through network shares. On some of these infections, a non-default configuration file was created, in which the newly infected machine configured itself to not use the external C&C server, but send all data to the infecting machine, therefore creating a peer-to-peer C&C infrastructure. Thus, all communication is passed through the infecting computer, making it possible for the attackers to even get access to devices that are not connected to the Internet.

The elegance of this technique is that it potentially provides access to secured zones within a network. Many secure networks have a 'secure' zone, in which internal (critical) servers are located. Hosts in this zone are better monitored and controlled than other hosts. The others hosts in the network form the 'insecure' zone, which are less well-protected. Although often hosts in the secure zone cannot directly be accessed from the external Internet, (some) hosts from the insecure zone can. Thus, creating a peer-to-peer C&C network may not be flagged up as malicious, even though effectively hosts in the secure zone are controlled from the external Internet. Additionally, as only one compromised computer in the network will connect directly to the C&C server, there is only one host showing suspicious traffic to outside the network.

Performing the attack When a machine is infected, the attackers were able to download additional modules to the malware, making it possible to use keyloggers, save screenshots, and steal documents and emails.

Covering tracks By default, Duqu has a lifespan of 30 - 36 days, after which it tries to erase itself from the machine. The C&C server could however extend this period.

3.2.2.2 Network data behaviour

A few observations on the network aspects of Duqu:

- As discussed before, the peer-to-peer option that Duqu supports is a smart way to make the APT successful. However, the very same technique is likely to result in deviating network patterns, as this means that there will be more traffic between the infected machine and its proxy.
- In case that a single host is proxy for one or more other machines, the host will exchange noticeable amounts of data to the C&C servers. This might be visible in the network data.
- The simple fact that infected machines are controlled manually from a C&C channel means that this may result in anomalous patterns, just as with the other examined APTs.

3.2.3 Flame

Although first detected in 2012, it is believed that Flame had been created five to seven years earlier [69]. Flame spied the target through a keylogger, by taking screenshots, intercepting emails, making microphone recordings, eavesdropping Skype calls and even connect through Bluetooth to nearby devices to download contact information. Flame infections took mainly place in the Middle East; it is believed the main goal of Flame was espionage, without targeting a specific industry.

An analysis of Flame showed that it infected USB drives in order to replicate, using the same two zero-day vulnerabilities used in Stuxnet. Beyond this, there are no signs that Flame is made by the same team behind Stuxnet or Duqu. Another way Flame propagated itself is through impersonating the Windows Update Server. In order to successfully do this, it performs a complex attack using an old certificate that relied on MD5 for its digital signatures, used in the Microsoft Windows operating system. Over 80 domains were used as command and control servers, which communicated over HTTP, HTTPS and SSH. Flame is able to recognise over 100 security products, and adopts itself depending on which products were found. Due to the highly advanced, sophisticated attacks part of the APT, experts have suggested the attack is state funded [69].

3.2.3.1 Attack phases

Reconnaissance It was analysed that Flame did not have a specific target in mind. Based on an analysis by Kaspersky, Gostev [29] report that there were no signs that Flame had a particular target, other than collecting intelligence, such as sensitive emails, documents and messages.

Compromising the target While it remains unclear how the initial infection was done, it is likely to have been through spear phishing or drive-by downloads.

Increasing foothold / lateral movement Flame can both replicate through hardware and through networks. Regarding the former, Flame can infect USB media such that upon connecting to a new (vulnerable) host, the host get infected as well. Regarding the latter, Flame can infect other computers in the network through printer sharing and remote job tasks. Also, if an infected host is a domain controller, it will create backdoor accounts on the hosts controlled by that host and infect those machines as well.

Using a highly sophisticated way to propagate itself through the network, Flame acted like a Windows Update server. As described by Symantec [65], a non-infected machine might use NetBIOS to query the network in order to find the WPAD server to get proxy settings. An Flame-infected machine will answer, claiming to be the WPAD server. All traffic from the non-infected machine will now be forwarded through the infected machine, serving as a proxy server. After this, when the machine tries to access Windows Update, the proxy will impersonate a Windows Update server and send a Flame loader (that appears to be signed by Microsoft) over Windows Update. The loader will be executed by Windows Update and subsequently infect the machine with Flame.

Performing the attack Flame's main goal is to gather sensitive information. To do so, Flame sniffs network traffic, takes screenshots, records audio from microphones and records key strokes using a keylogger. In addition, it is even capable of turning the infected computer into a Bluetooth beacon, trying to connect to near Bluetooth devices to look for sensitive information. The information is sent over the C&C channels over SSL on a regular basis.

Covering tracks A self-destruct command could be sent over the C&C channel to let Flame erase all its files and remove itself from the machine.

3.2.3.2 Network data behaviour

When considering how the network traffic patterns might change as a result of a Flame injection, one observes:

- Again, an obvious abnormality would be the communication to the C&C servers controlled by the attackers. Even though the communication is sent over SSL, the patterns of the C&C communication can possibly still be observed in the network traffic.
- The mechanism used by Flame to impersonate a Windows Update server certainly gives different network patterns. For instance, all traffic of a host is passed through another host acting as a proxy, for at least the period necessary to install Flame on the machine.
- Finally, in trying to get deeper into the system, Flame tries to infect other machines using the techniques described in the lateral movement phase. The methods that are performed over the network are likely to result in abnormal communication patterns: a host might contact another host that it usually does not connect to.

3.2.4 Carbanak

Carbanak [38] is an APT campaign discovered in 2015, mainly targeting financial institutions. It is estimated that between 500 million USD and 1 billion USD were stolen by the criminals through various means. In contrast to 'traditional' APTs, which focus on business-critical information, the goal of this APT was to steal money on a big scale, making Carbanak a cyber version of bank robbery.

3.2.4.1 Attack phases

Reconnaissance The main target of Carbanak was the financial sector. The report suggests that the attackers must have had insider knowledge, as Carbanak looks for tools that are mainly used within the financial world, such as tools to make large transactions.

Compromising the target The first penetration into the system was performed through spear phishing. The emails sent included malicious attachments that exploited vulnerabilities in Microsoft Office systems. Through the vulnerabilities, malware was installed that sets up a C&C channel. If certain banking applications are detected, special notifications are sent to the C&C server.

Increasing foothold / lateral movement According to the Kaspersky report [38], in most cases the target's network was compromised for between two and four months. Within this time, as many as hundreds of computers within a single target were affected. In this period of time, the attackers observed the behaviour of the users in order to learn about the normal routine. This information was then used to get access to the right victims and systems, and eventually to get the cash out.

As mentioned before, Carbanak enabled the attackers to study the workflow of users. Carbanak did so not only by traditional means such as keyloggers and taking screenshots, but it also video recorded users of the victim systems using video equipment such as webcams and security cameras. As a result of this, the attackers were able to observe how the systems were typically used, i.e. to understand the daily routine of their users in the target environment. This enabled the attackers to copy their behaviour in great detail without being noticed.

Performing the attack As a result of the in-depth analysis performed as part of the lateral movement stage, the attackers were able to select who to impersonate on the infected system. For instance, the attacker would insert fraudulent transactions to the internal database of the victim's financial software, making the attacker's movements hardly noticeable.

Additionally, sensitive information could be stolen as well. Classified emails, manuals, crypto keys and passwords were found on the C&C servers, giving the attackers internal knowledge about the systems of the targeted institutions.

Covering tracks The attacks stopped after $10 million was stolen. It does not follow from the report whether or how Carbanak covered its tracks after its mission was completed.

3.2.4.2 Network data behaviour

Carbanak is relatively hard to detect using network anomaly detection. A few notes:

- An obvious deviation would be the communication between the infected machines and the attacker's servers over C&C channels. Even though the communication itself was encrypted, the patterns of a C&C channel might still be observed in the network traffic.
- In the lateral movement phase, the attackers used common tools such as `Ammyy Admin` and SSH servers to control the victim's machine. As these tools may be usually accessed from within the internal network, seeing such traffic leaving the internal network may have been observed.

3.3 Attack characteristics

3.3.1 Attack metric

Having reviewed a variety of APT campaigns in section 3.2, it has become clear that existing APTs often have aspects that alter internal network traffic patterns. To illustrate this, a framework is constructed which lists the methods used by the investigated APTs that may cause different internal network traffic patterns. Using the APT life cycle (see section 2.1.2), various components are discussed.

3.3.2 Expand Access

A common tactic in order to expand the access within a network is to probe other nodes in the network in order to test whether they are vulnerable. There are various ways to perform this; some APTs seem to perform *de facto* a port scan, others try to see if hosts are vulnerable to specific vulnerabilities by querying a limited number of ports. Also, which hosts are queried varies: one could scan the whole network, a certain subnet or specific IP addresses. Harder to detect are APTs which study the behaviour of the infected host to see which hosts it usually connects to, and only queries these hosts.

In summary, when APTs are trying to expand their access, it is common to probe other hosts to see if they can be infected. As a result, new connections between hosts may appear. In table 3.1, these characteristics are listed as EA1 and EA2.

Table 3.1: Overview of observable characteristics in the internal network traffic, set out against the studied APT *campaigns.*

Description		Carbanak	Flame	Duqu	Stuxnet
Expand Access					
EA1	Probing all available hosts for vulnerabilities	-	✓	✓	✓
EA2	Probing known hosts for vulnerabilities	✓	-	-	-
Strengthen foothold					
SF1	Use peer-to-peer C&C (infected host as C&C proxy)	-	-	-	✓
SF2	Use peer-to-peer network to update threat	-	-	✓	-
SF3	Lead traffic over local proxy	-	✓	-	-
Perform attack					
PA1	Extract information: use of staging server	-	-	-	✓
PA2	Attack: insert malicious commands	✓	-	-	-

3.3.3 Strengthen foothold

The strengthen foothold phase could also cause observable changes in network patterns. Especially when the network consists of hosts that are not connected to the Internet, a common technique seems to set up peer-to-peer infrastructures to enable the attackers to get access to those machines. These infrastructures may, as seen in Duqu and Stuxnet, be used to give new instructions or update the malware. Additionally, for gathering intelligence or stealing credentials to get more foothold, a proxy infrastructure may be set up to eavesdrop on network data, as seen in Flame.

In other words, APTs may set up new communication infrastructures as a means to get to their target. These infrastructures may result in new connections and different flow sizes. These characteristics are referred to as SF1, SF2 and SF3 in table 3.1.

3.3.4 Perform attack

Depending on the goal of the attackers (such as stealing information, damaging/destroying infrastructure, performing malicious actions), an APT may cause abnormal network patterns. When the goal is to steal intelligence, the APT may transfer the information through the network, for instance when the information comes from a host that has no connection to the Internet. Similarly, when the goal is to perform malicious actions, the commands sent by the attacker may result in uncommon patterns.

Thus, depending on the goal of the attackers and the network infrastructure, APTs may alter the internal network patterns as well. These types of attack are listed as PA1 and PA2 in table 3.1.

Figure 3.2 provides some examples from the characteristics of table 3.1, visualising how the characteristics may result in observable, anomalous behaviour. Note that these are simplifications; in reality, the behaviour caused by APTs may not be as obvious as in the given situations.

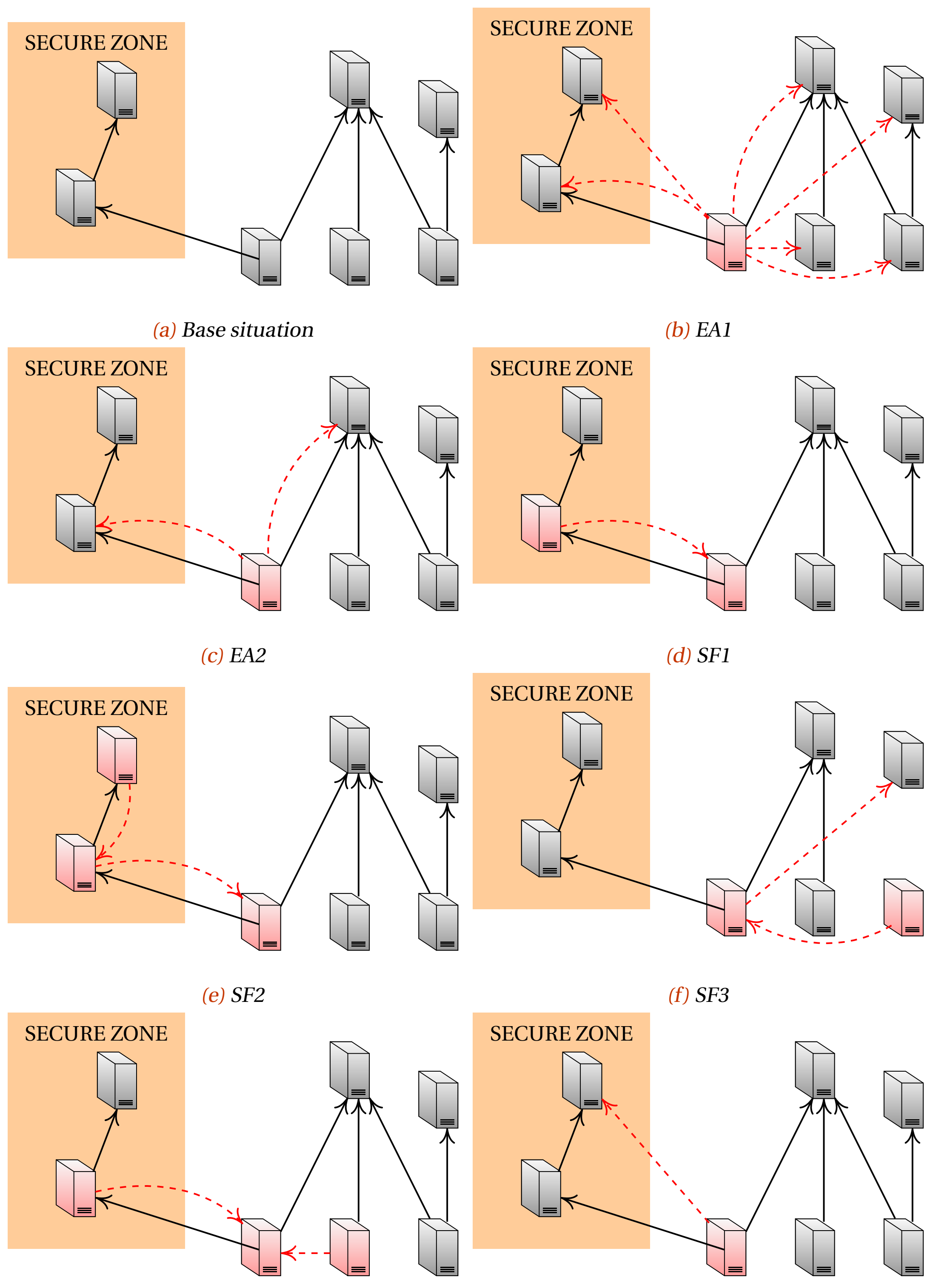
3.4 Conclusion

After having analysed four major APT campaigns and having identified their characteristics, it is possible to make some statements about the relationship between APT campaigns and internal network traffic patterns.

A conclusion that can be drawn from the analysis is that the reviewed APT campaigns are indeed unique in their methods. As a consequence, misuse-based approaches are highly unlikely to detect this type of attack.

Still, it is possible to identify similarities between the underlying approach taken by the attackers, as there are steps that the attacks share. For each of the attacks, it is possible to identify the actions taken in the various APT life cycle stages. Thus, despite their uniqueness, it possible to make higher level statements about APT behaviour.

Another observation is that identified characteristics confirm the hypothesis that there is a strong internal network traffic aspect in APT campaigns. For three life cycle stages, we have shown that an APT intrusion might lead to abnormal networking patterns. Supported by this finding, a new opportunity for APT detection by monitoring and analysing internal network patterns can be justified. The question that now arises is how these characteristics ought to be used to detect network intrusions by monitoring internal network traffic.



(a) *Base situation*

(b) *EA1*

(c) *EA2*

(d) *SF1*

(e) *SF2*

(f) *SF3*

(g) *PA1*

(h) *PA2*

Figure 3.2: *Examples of how internal network structures may change as the result of an* APT *intrusion. Hosts in the secure zone do not have external Internet access, hosts outside the secure zone have. Red hosts indicate infected hosts, red arrows indicate observable, changed behaviour caused by the corresponding* APT *characteristic.*

4 Network host clustering

To investigate a new method for network intrusion detection based on clustering hosts and analysing their behaviour, it is necessary to take a closer look at clustering techniques. Section 4.1 discusses clustering from a data analytics point-of-view, describing various clustering algorithms and the differences between them. This analysis is used in section 4.2 to apply clustering to the domain of network host clustering. The same section also investigates which requirements a clustering method should satisfy for application in this domain and evaluates the studied clustering methods against these requirements.

4.1 Clustering methods

In data analytics, it is useful to identify patterns in a given set of data for analysis of other (similar) data. In other words, given training data, one wants to come up with a pattern that helps predicting the behaviour of (unseen) test data, or one wants to determine whether (unseen) test data is part of the same category as the training set. The process of recognising a pattern, called learning, is divided into *supervised* learning and *unsupervised* learning, also known as respectively *classification* and *clustering* [36]. The latter is considered to be harder than the former; however, there is also a hybrid learning variant which is partly supervised, partly unsupervised.

The main goal of data clustering is to "discover natural grouping(s) of a set of patterns, points, or objects" [36, p. 652]. In literature, there are many ways to determine such natural grouping(s). In the next subsections, some clustering[2] techniques that are regularly used within network data analysis, are discussed.

4.1.1 Selecting clustering algorithms

Within the clustering science, there are numerous clustering methods available, all using different algorithms that may result in different clusterings for the same input data. Research by Jain et al. [37] compared 35 different clustering algorithm and were able to show that these algorithms can be divided into three groups, giving similar results within these groups. The taxonomy proposed by the authors can be found in figure 4.1.

[2]We distinguish:

- A *clustering* is a grouping of a set of patterns, points or objects;
- A *cluster* refers to a single group within a clustering;
- A *clustering method /technique /algorithm* is the method applied to derive the clustering.

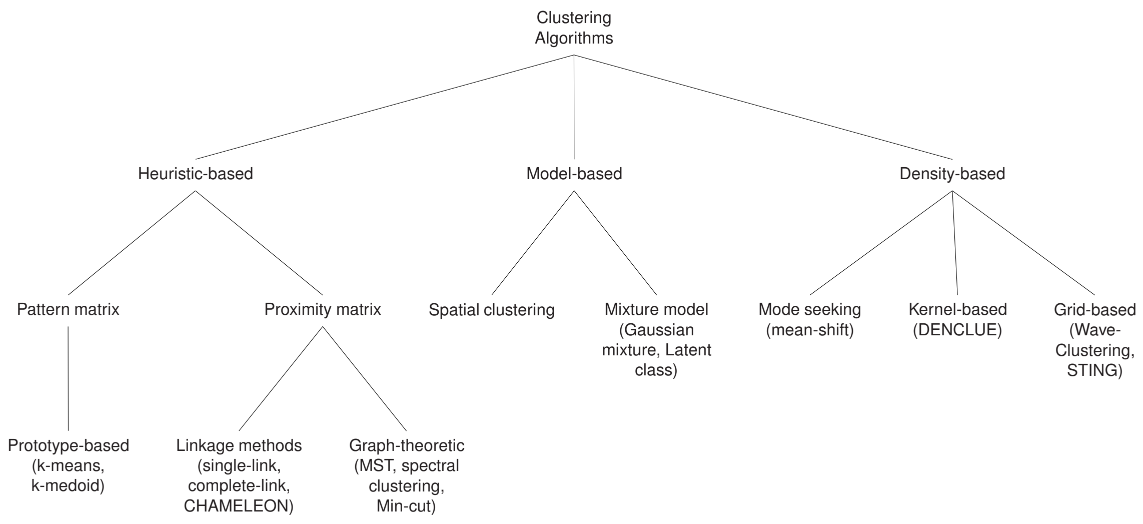

Clustering Algorithms
Heuristic-based
Model-based
Density-based
Pattern matrix
Proximity matrix
Spatial clustering
Mixture model (Gaussian mixture, Latent class)
Mode seeking (mean-shift)
Kernel-based (DENCLUE)
Grid-based (Wave-Clustering, STING)
Prototype-based (k-means, k-medoid)
Linkage methods (single-link, complete-link, CHAMELEON)
Graph-theoretic (MST, spectral clustering, Min-cut)

Figure 4.1: Possible taxonomy of clustering methods according to Jain et al. [37].

The authors conclude that there is no such thing as a best clustering algorithm. They do however assert that good partitions are formed when there is a solid match between model presumed by the clustering method and data. Thus, as the structure can in most cases not be known *a priori*, one should try multiple methods to find an appropriate clustering algorithm for a set of data [36].

Based on this notion, a selection of available clustering methods will be evaluated, from various categories using different types of input data.

As there are too many clustering methods available to compare at once, finding the best clustering algorithm or clustering algorithm implementation is outside the scope of this research. However, for the purposes of this research, it will be analysed which class of clustering methods performs best when it comes to detecting anomalies by comparing clustering methods from these classes.

Thus, the investigated clustering methods have been selected based on their class and their input data. Table 4.2 provides an overview of the selected clustering methods, which will be studied in greater detail in the following sections.

Table 4.2: Overview of the reviewed clustering methods, with their classes and type of input.

Clustering method	**Class**	**Type**
k-means	Heuristic-based → Pattern-based	Vector quantisation
Mean shift	Density-based → Mode seeking	Vector quantisation
Louvain	Density-based	Graph-based
SBM	Model-based	Graph-based

4.1.2 k-means clustering

k-means [45] is a popular, relatively simple clustering algorithm used in a wide spectrum of scientific fields [36]. The algorithm relies on the distance between values of the entities in order to divide a group of entities into k clusters.

For instance, Münz et al. [48] apply k-means clustering to the field of network traffic analysis by using the algorithm with flow records in order to distinguish normal traffic from anomalous traffic. As feature space, the number of packets, total number of bytes and number of different source-destination pairs are taken.

4.1.2.1 Algorithm

Let $X = \{x_i\}$ with $i = 1,...,n$ be a set of n d-dimensional vector spaces. Each point represents an entity to be clustered, expressed in d characteristic values. Let $C = \{c_j\}$ with $j = 1...k$ be the set of clusters. Let μ_p be the mean of cluster c_p. The algorithm now tries to find the minimum sum of squared error over all k clusters, i.e. $\sum_{i=1}^{k}\sum_{x\in c_i}\left\|x-\mu_i\right\|^2$. For this, a distance function[3] is used.

The k-means clustering algorithm consists of one initial step and one iteration step. The iteration step is repeated until the input is the same as the output.

Initial step Randomly select k points from X as initial means. In other words, μ_i is set to the ith selected point (for $0 \le i < k$).

Iteration step The iteration step consists of the two following actions:

1. Assign each point to the closest cluster centre μ_i, by evaluating the distance function for each $x \in X$ to each μ_i.
2. Compute the new cluster centre μ_i as the middle of all points assigned to μ_i.

This is repeated until the algorithm converges.

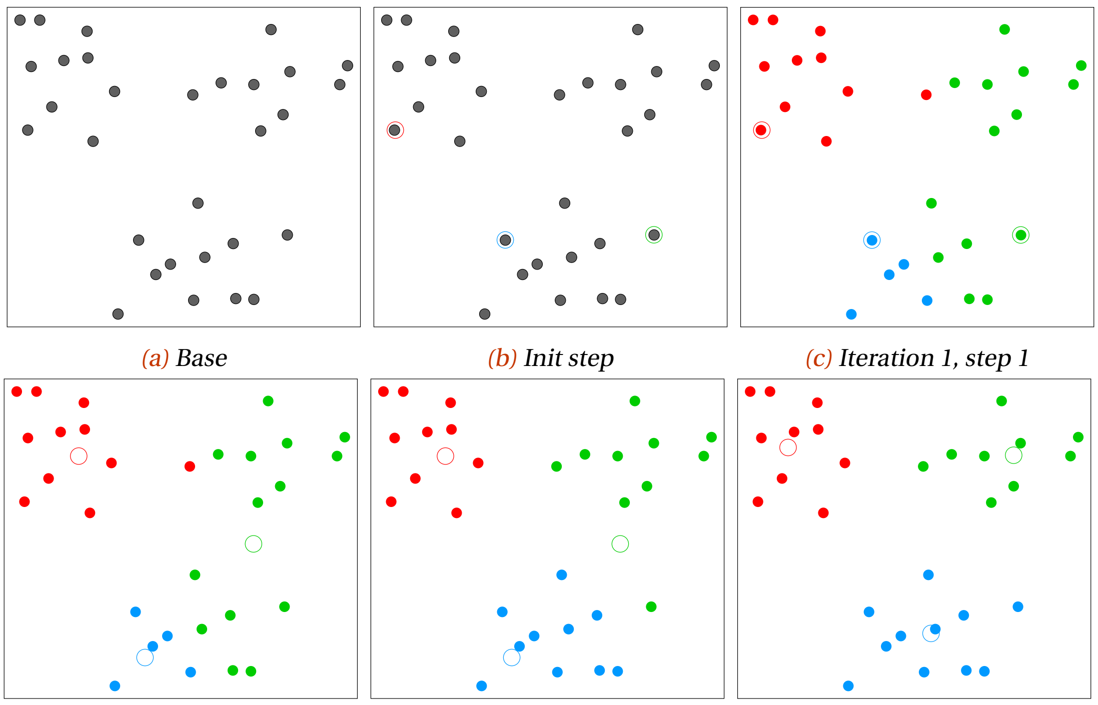
4.1.2.2 Variants

There are various algorithms similar to or based on k-means. Two of them are described in this section.

Fuzzy c-means As an extension to k-means, fuzzy c-means is a *soft* algorithm, meaning a single point can belong to multiple clusters. Again, the goal is to find the minimum sum of squared error over all clusters, i.e. $\sum_{i=1}^{k}\sum_{x\in c_i} w_i(x)\left\|x-\mu_i\right\|^2$. The algorithm equivalent to k-means, except that the cluster means μ_k are now calculated using the weights, giving $\mu_i = \frac{\sum_{x\in X} w_i^m(x)x}{\sum_{x\in X} w_i^m(x)}$ and define the weights as $w_i^m(x) = \frac{1}{\sum_{j=1}^{k}\left(\frac{\|x-\mu_i\|}{\|x-\mu_j\|}\right)^{\frac{2}{m-1}}}$. Parameter $m \in \mathbb{R}$ (with $m > 1$) indicates the level of cluster fuzziness: the bigger the value of m, the smaller the number of memberships $w_i(x)$.

x-means To overcome the issue with not knowing the number of clusters in advance, x-means [57] applies an information criterion to determine whether or not a group of entities

[3]Often, the Euclidean distance is used. The Euclidean distance is defined as $d(x_i,\mu_j) = \sum_{x_i\in c_j}\left\|x_i-\mu_j\right\|$.



(a) *Base* (b) *Init step* (c) *Iteration 1, step 1*

(d) *Iteration 1, step 2* (e) *Iteration 2* (f) *Final iteration*

Figure 4.3: An example of running the k-means algorithm using a two-dimensional feature space.

should be split into separate clusters or not. Often the Bayesian Information Criterion (BIC) or the Akaike Information Criterion (AIC) are used for this purpose.

The first step of the algorithm is to run the k-means algorithm, initially with a low value of k such as 2 or 3. After that, the second step is to split up all centroids into two children, followed by running k-means locally (i.e. with $k = 2$) on the original cluster using the generated two children as centroids. After the algorithm converged, it is tested using the information criterion whether the newly created clusters indeed are modelling real structure. If this is the case, the two children remain; otherwise, the parent remains.

These two steps are repeated until either the second step does not bring new cluster or if an upper bound k_{max} is reached.

4.1.3 Mean shift clustering

Mean shift [15] is a density-based clustering algorithm, meaning that it tries to find the maximum of a density function. In contrast to k-means, no prior knowledge of the number of clusters is required.

Mean shift is often applied in the field of image processing. Comaniciu et al. [16] describes how mean shift can be used for real time tracking moving objects using cameras. Beyond this, it can be applied to a wide range of cases, as it only requires multidimensional variables.

4.1.3.1 Algorithm

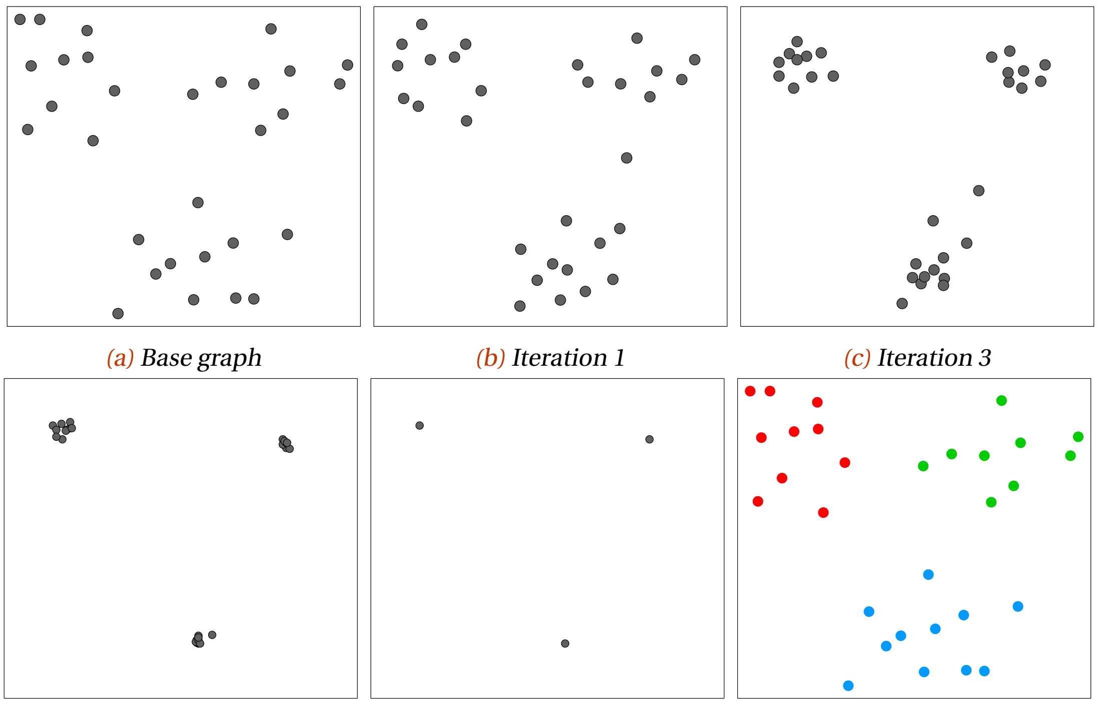
Let $X = \{x_i\}$ with $i = 1, ..., n$ be a set of n d-dimensional points. Each point represents an entity to be clustered, expressed in d characteristic values. The kernel density function is defined as $f(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$ with h the window radius. The Radial Basis Function (RBF) kernel $K(x)$ determines the weight of points nearby, used for re-estimation of the mean. K can often be defined as $K(x) = c_{k,d}\, k(|x|^2)$. Function $k(x)$ represents the kernel profile function, for which often the Gaussian kernel $k(x) = e^{-\frac{x}{2}}$ is used. $N(x)$ is defined as the *neighbourhood* of x, a set of points for which we have $K(x) \neq 0$.

Initial step Select all points $x \in X$.

Iteration step The iteration step consists of the two following actions for each x:

1. Compute the main shift vector $m(x) = \frac{\sum_{x_i \in N(x)} K(x_i - x) x_i}{\sum_{x_i \in N(x)} K(x_i - x)}$.
2. Translate the window $x = x + m(x)$.

This is repeated until the algorithm converges, which is guaranteed to happen [15].



(a) Base graph (b) Iteration 1 (c) Iteration 3 (d) Iteration 6 (e) Iteration 9 (f) Projection of clustering on original graph

Figure 4.4: An example of running the mean shift algorithm using a two-dimensional feature space.

4.1.4 Louvain community clustering

Community clustering [25] is recognising groups in a connected network of nodes. Unlike the aforementioned types of clustering, community clustering is part of graph theory.

Blondel et al. [5] present the Louvain method, an algorithm that optimises modularity in graphs. In other words, the resulting clustering is based on the level of communication between the hosts. An advantage of the Louvain method is its ability to scale efficiently to larger data sets. The authors estimate the method has a complexity of $O(n \log n)$, although they did not verify this.

In network data science, community clustering methods are useful for creating clusters (in this context called *communities*) of nodes that communicate with each other. In the original research of Blondel et al. [5], the Louvain method is applied to a Belgian mobile phone network consisting of 2.6 million nodes and 6.3 million edges. The researchers were able to show a hierarchy of six levels in the phone network, each level being one pass of executing the two steps of the algorithm.

4.1.4.1 Algorithm

A requirement of the algorithm is that the number of edges m should be of the same order of the number of nodes n of the graph. If this requirement cannot be satisfied, the distribution of edges among the nodes is too homogeneous to make useful groupings.

The algorithm relies on *modularity*, a measure for the density of links inside communities compared to links between communities. Modularity is defined as

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j) \tag{4.1}$$

with $Q \in \mathbb{R}, -1 \leq Q \leq 1$. A_{ij} represents the weight between nodes i and j, k_x is the sum of the weights of edges attached to node x, m is half the sum of all edge weights in the graph, c_x is the community of node x and δ is a simple delta function.

The algorithm starts with a weighted network of n nodes.

Initial step Each node in the graph is assigned to its own community. Hence, the algorithm starts with n communities.

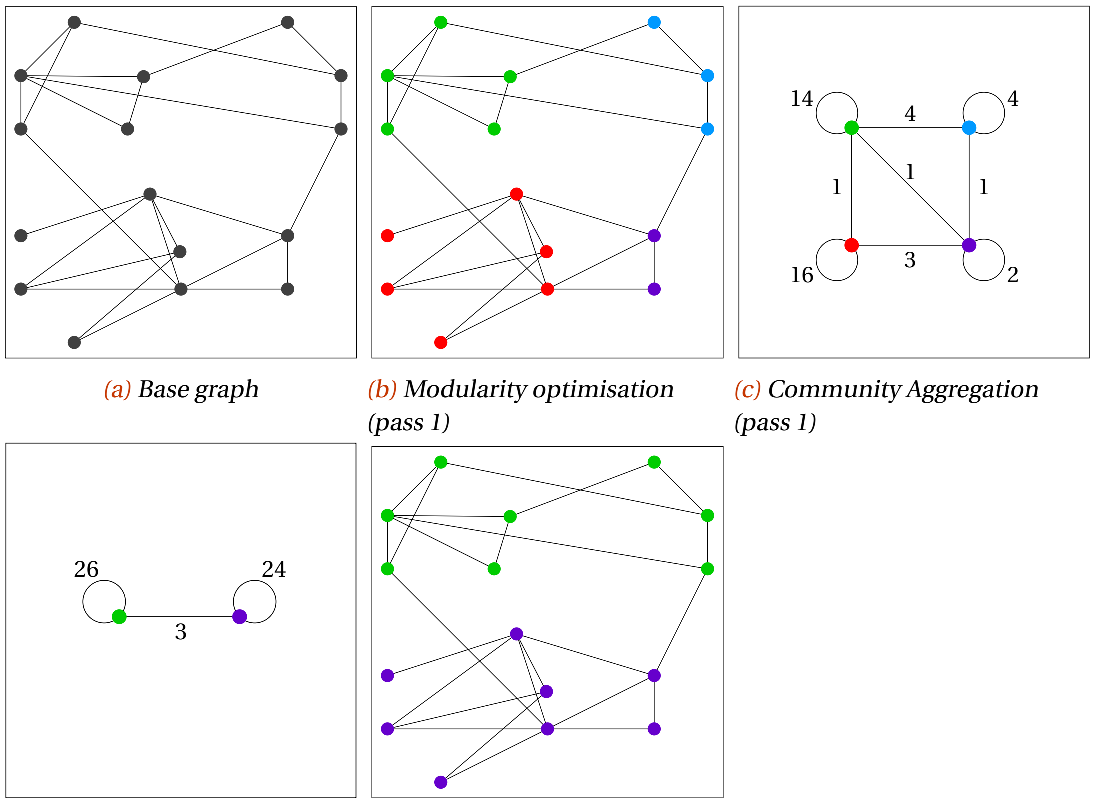
Part 1: Modularity optimisation For each node i in the graph, consider each neighbour j. Evaluate the gain in modularity in case i was removed from its community and by placing it in the community of j. If there is a community for which the gain is positive and maximal, the node will be placed in that community. If the maximum gain is negative, i will remain in the same community.

This process is repeated for all nodes until the algorithm converges.

Part 2: Community aggregation Based on the communities found in part 1, a new graph is built where each community is represented as one node. Links between nodes of the same community are represented as a (weighted) self-loop on the new community node, while links from nodes from one community to another is represented by weighted edges between the new community nodes. After the new graph is created, part 1 may be applied again.

4.1.4.2 Variants

There exist several variants within graph community clustering [58]. These can be broadly divided into minimum-cut, hierarchical, Girvan-Newman, modularity, statistical interference



(a) Base graph

(b) Modularity optimisation (pass 1)

(c) Community Aggregation (pass 1)

(d) Pass 2

(e) Projection of second pass on original graph

Figure 4.5: An example of running the Louvain community clustering algorithm.

and clique-based methods. Some of these alternative community clustering methods are discussed in this section.

Minimum-cut approach A basic approach towards the problem of graph clustering is the minimum-cut method. The network is divided into a (predetermined) number of communities, such that the number of edges between the groups is minimal. Because the number of communities is fixed beforehand and because it does not take the implicit network structure into account, this method is not often applied in network science.

Bron-Kerbosch algorithm The Bron-Kerbosch algorithm is a clique-based method for finding maximal cliques. A clique is a subset of a graph of which its induced subgraph is complete, i.e. every two distinct vertices in a clique are adjacent. Based on the cliques found by the algorithm, one may define communities as all cliques that have overlapping vertices/edges. An even simpler approach would be to take all cliques with a size bigger than one as a community in itself.

Girvan-Newman algorithm Girvan and Newman [28] propose a graph clustering method that relies on vertex betweenness, a measure for how central nodes are in a network. The algorithm finds the edge with the highest betweenness, removes the edge, and recalculates the betweenness for all remaining edges. This step is repeated until no edges remain. The algorithm was included in the Java Universal Network/Graph Framework (JUNG)[4]

[4]See http://jung.sourceforge.net/.

and the Stanford Network Analysis Project (SNAP)[5], helping the algorithm becoming increasinlgy popular. A major disadvantage of the Girvan-Newman algorithm is its high complexity, $O(m^2 n)$ for m edges and n vertices, meaning the algorithm is inefficient for larger networks.

4.1.5 Stochastic Blockmodel Clustering

The concept of stochastic blockmodels (SBM) was first presented by Holland et al. [33]. Stochastic blockmodels are random graph ensembles in which vertices are assigned to discrete groups called *blocks*. The probability of an edge existing between two vertices is determined according to the block it belongs to.

Stochastic blockmodels rely on the principle of generative models, which assumes that the given graph was generated according to a model θ. Given θ, it is possible to draw a graph G from the distribution. In the context of clustering, however, one does the opposite: given G, choose a θ that makes G likely. See also figure 4.6. As blocks are a variation on communities, we generally have $p_i > p_j$ with p_i the probability that that two vertices in the same block are connected and p_j the probability that two vertices in different blocks are connected.

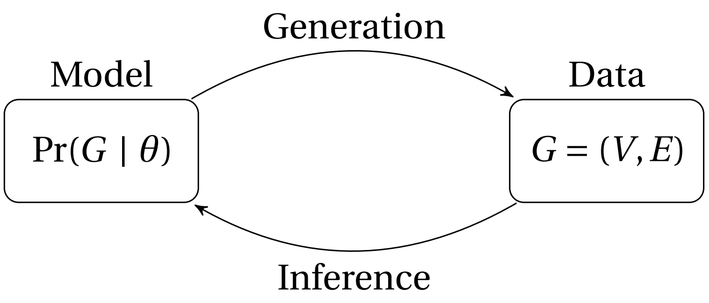


Figure 4.6: Relation between model and observed data according to the Stochastic Blockmodel principle.

Stochastic blockmodels can also be used to detect community structures in networks [49]. SBM clustering is an extension of random graphs, in which any two nodes are linked independently with a certain probability. SBM clustering is characterised by the fact that each vertex belongs to a hidden cluster, and that connection probabilities between vertices depend exclusively on their (unobserved) clusters. In other words, by inferring the graph a clustering can be derived.

4.1.5.1 Algorithm

Consider an undirected, unweighted graph G, consisting of N vertices. Assume the number of blocks B is fixed.

It is assumed that the given graph G is generated according to the following scheme:

$$\begin{aligned} z \mid \theta &\sim \text{Multinomial}\left(1, (\theta_1, \ldots, \theta_B)\right) \\ \eta(a,b) \mid \beta &\sim \text{Beta}\left(\beta, \beta\right) \\ R(i,j) \mid z, \eta &\sim \text{Bernoulli}\left(\eta(z_i, z_j)\right) \end{aligned} \tag{4.2}$$

[5]See http://snap.stanford.edu/.

where Multinomial($\mathbb{N}, \mathbb{R}^B$) represents a multinomial distribution such that θ_i is the probability of a node being assigned to cluster i. $\eta(a,b)$ the probability that a link exists between an entity of cluster a to an entity of cluster b. R indicates the relations between entities, and can therefore be seen as an adjacency matrix. In other words, $R(i,j) \to \{0,1\}$ indicates whether the entities i and j have a link or not.

In order to derive the clustering, inference is required. Given the observed relations (R), the underlying model (z) is inferred by finding a z that maximises

$$\begin{gathered}\Pr(z \mid R) \propto \Pr(R \mid z) P(z) \\ \text{with } P(R \mid z) = \Pi_{a,b \in [1,B]} \frac{\operatorname{Beta}\left(m_{ab} + \alpha, \bar{m}_{ab} + \beta\right)}{\operatorname{Beta}\left(\beta, \beta\right)}\end{gathered} \tag{4.3}$$

where m_{ab} is the number of 1-edges between classes a and b (i.e. $R(i,j) = 1$), $\bar{m}_{ab}$ is the number of 0-edges between classes a (i.e. $R(i,j) = 0$).

This can for instance be achieved by using Markov chain Monte Carlo (MCMC) [55], by trying to move objects from one cluster to another, mering clusters of splitting one. Other inference methods include expectation propagation and mean-field variational methods [1].

The obtained z represents the underlying model used for generating the graph, hence providing the supposed clustering.

4.1.5.2 Variants

The SBM clustering algorithm comes in many flavours, such as binomial SBM, mixed-membership SBM, hierarchical SBM, fractal SBM, infinite relational model, degree-corrected SBM, bipartite SBM, and many more. Depending on the task, it can be useful to use a variation on the original algorithm. Three of SBM's common variations are:

Degree-corrected One of the limitations to the original SBM is that it assumes all nodes within a community are stochastically equivalent. In networks with highly varying node degrees, this may result in clustering by degree and can as a result give a poor fit. By correcting for degrees in SBM, this can be overcome [71].

B-finding Another issue is that the original SBM algorithm assumes a fixed number of blocks B. One approach is to estimate the maximum number of blocks that are expected to represent a useful clustering (i.e. not $B = n$) by taking the blockmodel entropy into account [54]. An approach to derive an optimal value for B is using MCMC simulations [55].

Mixed membership For some data sets, single entities might not necessarily belong to a single cluster, but can have concurrent cluster memberships. The mixed-membership version of SBM [1] assigns for each entity a probability of belonging to each cluster.

4.1.6 Evaluating clusterings

Given that there are various ways to cluster a given network, it is desirable to evaluate to what extent different clustering methods result in similar clusterings. In other words, a measure for the degree of overlap (or 'agreement') between two clusterings of the same graph. In literature, there are several methods for comparing two clusterings, of which the Rand index and the adjusted Rand index are often used.

Consider a set S, containing n elements, and two clusterings $X = \{X_1, ..., X_r\}$ and $Y = \{Y_1, ... Y_s\}$ which are partitionings of S into respectively r and s subsets.

The *Rand index* [59] is a measure of similarity between two data clusterings. It can be used to compare differences in two types of data clustering, or to evaluate the performance of a clustering by comparing it to a predefined (i.e. manually defined) clustering. It is defined as

$$RI = \frac{a+b}{\binom{n}{2}} \tag{4.4}$$

where a is the number of pairs in S that are in the same set in X and in the same set in Y, and b the number of pairs in S that are in different sets in X and in different sets in Y. The Rand index is a value between 0 and 1 in which 1 indicates a perfect match whilst 0 means the clusterings have nothing in common.

As an extension of the standard Rand index, the *adjusted Rand index* (ARI) [34] takes probabilities into account. It solves the issue with the standard Rand index that gives a non-constant expected value for two random partitions. The adjusted Rand index is corrected for the possibility that two partitions agree through chance alone.

To compute the ARI, it is necessary to compute a contingency table first. This two-dimensional table compares each cluster from the first clustering to each of the clusters from the second clustering. We define n_{ij} as $|X_i \cap Y_j|$, i.e. the number of elements that cluster i from clustering X and cluster j from clustering Y share. Furthermore, $a_i = \sum_j |X_i \cap Y_j|$ and $b_j = \sum_i |X_i \cap Y_j|$.

The adjusted Rand value is defined as

$$ARI = \frac{Index - ExpectedIndex}{MaxIndex - ExpectedIndex} \tag{4.5}$$

with $Index = \sum_{ij} \binom{n_{ij}}{2}$, $MaxIndex = \frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right]$ and $ExpectedIndex = \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}$. Variables a_i, b_j and n_{ij} are values from the contingency table.

The value produced is bounded above by 1, while in contrast to the standard Index it can take negative values. A negative score indicates dissimilarity between the two clusterings, while a positive score indicates similarity (with 1 indicating a perfect match). Comparing a random clustering to another clustering will give the ARI an expected value of 0.

4.2 Host clustering

Having studied various clustering methods, it will be investigated in this section which clustering methods are most suitable for clustering hosts in an internal network, with the ultimate goal to improve anomaly detection. First, it is investigated why it is worthwhile to cluster hosts. Secondly, a set of requirements which a clustering method should satisfy are discussed in section 4.2.2. In the section that follows, a systematic validation of the requirements for each of the aforementioned clustering methods is given (section 4.2.3). Finally, a conclusion on clustering methods in the context of internal network traffic is given in section 4.2.4.

4.2.1 Rationale

4.2.1.1 Background

As has been already discussed in the problem statement (section 1.1), a common issue with network traffic anomaly detection is the high false positive rate, making the manual process of evaluating events (i.e. the incidents reported by the detection system) inefficient and cumbersome. This is very unfortunate, because as discussed in chapter 2, anomaly detection is a useful additional asset in detecting advanced network intrusions. To overcome this issue, it is necessary to get a better understanding of what causes these high false positive rates. With this knowledge, it might be possible to improve the *status quo* into something more flexible and efficient towards detecting advanced network intrusions.

Model-based anomaly detection in particular suffers from the issue that it relies on human behaviour: the underlying assumption is that it is possible to be capture the behaviour of a user in a (mathematical) model, describing the bounds in which a machine or a user under normal circumstances behaves. However, there is a strong case for the claim that this premise does not hold.

First of all, uncertainty is inherent in predicting human behaviour. Although it is true that most users tend to work in a routine, the human mind is too complex to capture in a simple model. Users' unpredictable behaviour inevitably increases the number of false positives, as even small, harmless anomalies are abnormalities. Secondly, the notion of 'normal circumstances' is dubious. It is challenging for an anomaly detection system to take all external factors into account that might impact the monitored situation. For example, in a business setting, a public holiday might lead to unexpected networking patterns; a developing news story might virtually bring job-related network traffic to a halt, while traffic to news websites peaks. Even an employee taking a day off or working an extra day can be seen as anomalous. Although such abnormalities can be seen as anomalous, they are not necessarily harmful and therefore increase the false positive rate.

In conclusion, the crux of the described problem seems to be the inability to create a reliable model of what should be seen as 'normal' behaviour. Overcoming this issue is not self-evident: as anomaly detection depends on human actions, it will be very difficult to create a system with a consistent false positive rate of 0% and a true positive rate of 100% for the aforementioned reasons. Thus, an effective anomaly detection system without human interpretation and intervention seems to be a utopia.

One way to have the advantages of anomaly detection while reducing the false positive rate, is by taking a multi-layered approach in advanced intrusion detection. This involves using multiple mechanisms in parallel and correlating the results to identify threats. The underlying idea is that by monitoring multiple aspects of the environment from multiple perspectives (e.g. host-based and network-based, signature-based and anomaly-based, etc.), true positives are likely to be detected by a majority of the mechanisms while false positives are more likely to be detected by single or a few mechanisms.

For this to work, the underlying, individual mechanisms should produce reliable results, as this will increase the overall results. As anomaly detection currently is a weak link in this context, a central question regarding anomaly-based intrusion detection is whether and how the false positive rate can be reduced while not sacrificing the true positive rate.

4.2.1.2 Role of clustering

We propose an improvement to the existing anomaly detection mechanisms by applying host clustering in the anomaly detection process. As has become clear from the first part of this chapter, it is possible to create groupings of hosts that are in some respect similar. This aspect of similarity might be a valuable contribution in creating a wider understanding of user's behaviour.

As noted in the previous section, the behaviour of users follows to some degree regular patters, where small, harmless deviations from these patterns are likely to cause higher false positive rates. Clustering might overcome this phenomenon by taking the 'average' behaviour of similar hosts together. Since the hosts in a single cluster are in some respect alike, it can be argued that the differences in behaviour between the hosts are 'acceptable', and hence fall within the bounds of normal behaviour. Using this 'common' or 'average' behaviour of the clustered hosts, it might be possible to create a more accurate definition of what normal behaviour is for the hosts concerned.

Following this line of reasoning, this approach will reduce the number of false positives, as what might be seen as an abnormality for an individual host might not be an abnormality for hosts of the same cluster, hence not marking such events as anomalous. Yet, it is expected that it will not reduce the true positive rate, as true abnormalities are expected to be anomalous compared to the cluster's common behaviour as well.

Be that as it may, this theory only holds when the applied clustering algorithm results in a clustering that indeed groups together hosts that are similar in a way that is relevant for anomaly detection. Hosts may be similar in various ways, not all necessarily relevant for anomaly detection. The purpose of the remainder of this chapter is to evaluate what clusterings the introduced clustering methods yield when applied to network traffic data and how these relate to anomaly detection.

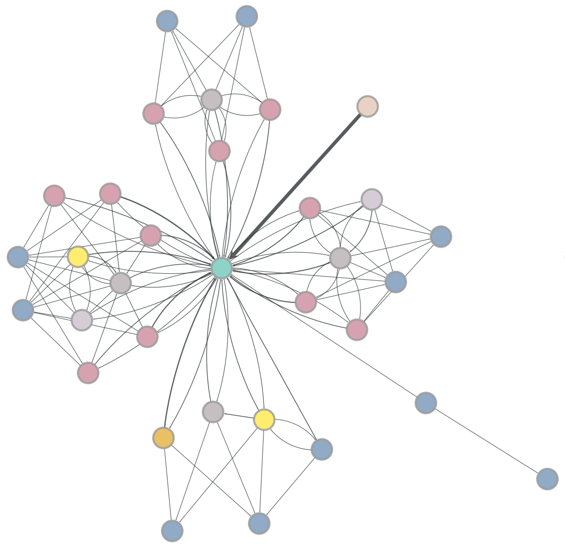
4.2.1.3 Applying clustering to network data

As set out in section 4.1, k-means, mean shift, Louvain Community Clustering and Stochastic Blockmodel clustering will be evaluated in the context of internal network traffic. This work will focus on the standard algorithms of the described four clustering methods.

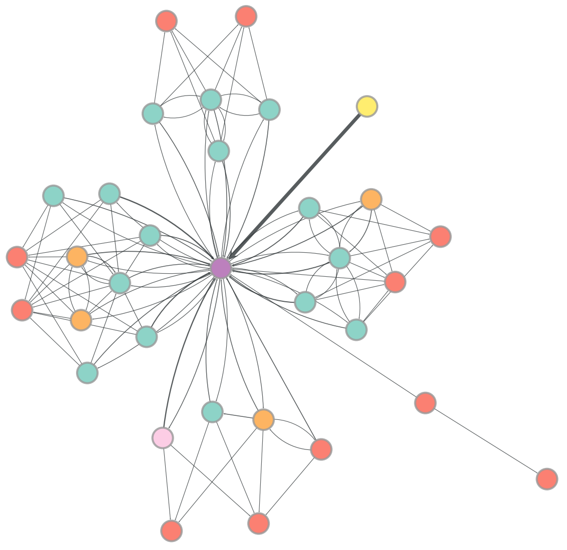
To illustrate the differences of these clustering methods, a comparison between them is provided in figure 4.8 and table 4.9 for a simple, small network consisting of 33 hosts. Connections are here defined as flow between the hosts, i.e. an edge indicates that there has been at least one flow between the hosts. The weight of the edges increases with the number of flows.

Table 4.10 shows the ARI scores between the four clusterings. The comparison shows that k-means and mean shift are somewhat similar as are Louvain and SBM. This is likely to be due to the scope of the clustering methods (vector quantisation versus graph-based), in which the two combinations are similar.

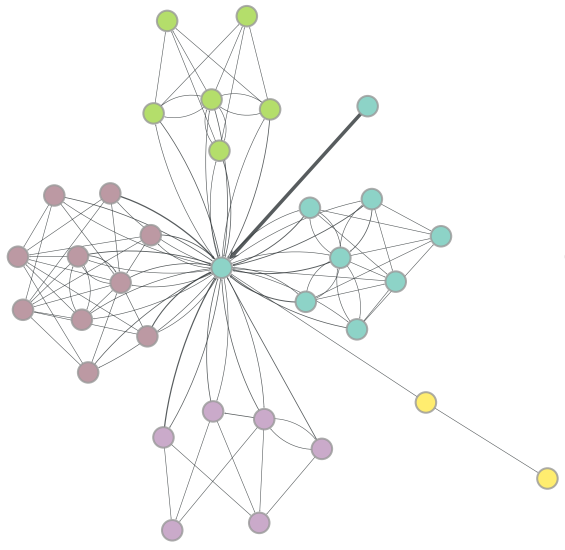
What follows from this is that the obtained clusterings are different and provide a different grouping of the same network. In order to make statements about which type is most suitable for our purposes, the clustering methods will be assessed on a number of requirements.



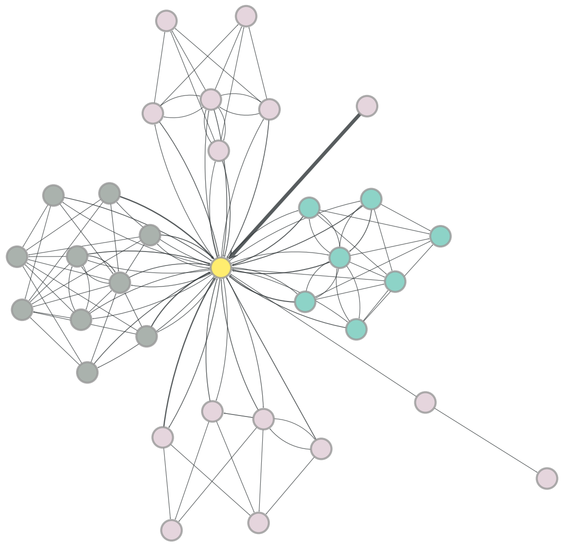
(a) k-means clustering



(b) Mean shift clustering



(c) Louvain clustering



(d) Stochastic Blockmodel clustering

Figure 4.8: Comparison of clustering techniques on the UNB ISCX *2012 data set*[6] *of k-means, mean shift*[7]*, Louvain Community Clustering and Stochastic Blockmodel clustering.*

Table 4.9: Overview of the partitioning per clustering method with their corresponding colour and size in figure 4.8.

Method	**N/o clusters**	**Clustering**
k-means	8	1 , 11 , 11 , 1 , 1 , 2 , 4 , 2
Mean shift	6	15 , 11 , 4 , 1 , 1 , 1
Louvain	5	9 , 10 , 6 , 6 , 2
SBM	4	7 , 10 , 15 , 1

[6]This set is evaluated in section 6.1. For this comparison, data from Saturday 12 June 2012 was used.
[7]For both k-means and mean shift, the used feature set comprises:

$$[\text{COUNT}(Incoming\_Connections), \text{COUNT}(Outgoing\_Connections), \text{AVG}(Duration_{In}), \text{AVG}(Duration_{Out}), \text{AVG}(Bytes_{In}), \text{AVG}(Bytes_{Out}), \text{AVG}(Packets_{In}), \text{AVG}(Packets_{Out})]$$

.

Table 4.10: ARI *scores between the clusterings of figure 4.8.*

	k-means	Mean shift	Louvain	SBM
k-means	1.000	0.771	-0.0226	0.0168
Mean shift	-	1.000	-0.00204	0.0108
Louvain	-	-	1.000	0.577
SBM	-	-	-	1.000

4.2.2 Requirements

As follows from section 4.1, there are various ways to cluster data. Depending on the information available and the goal of the clustering, one can choose a clustering method that gives the best results.

As for this research it is investigated how clustering hosts in an internal network environment with as ultimate goal to detect anomalies in an effective and efficient way, it is possible to derive a list of requirements a clustering method should satisfy. Even though some assumptions may seem evident, it is important to define the underlying assumptions for our context.

As has been argued in section 4.2.1, hosts in the same cluster should be similar in such a way that it is possible to create a model of normal behaviour per cluster. Even though there are other aspects on which it is possible to cluster hosts, our focus is internal network behaviour.

Additionally, the fact that model-based intrusion detection inherently relies on comparing training data to test data has implications for the most suitable clustering method. In order to create reliable and verifiable anomaly detection, it is important that the clustering method gives consistent results. In other words, it is necessary that the clustering method behaves (at least somewhat) deterministic. If a clustering method would result in completely different clusterings each time it is run, it is less likely the the resulting clusterings are actually representing anything, as they appear to be interchangeable.

Furthermore, as network environments are dynamic in nature, it is important that the used clustering method has some tolerance for changes, for similar reasons as the previous requirement. The extent to which a clustering method remains stable as a result of minor changes is a measure of the quality of the clustering. In other words, clustering methods which prove to be volatile are less suitable for application to the networking domain.

Finally, it is important that the clustering method is able to perform well under vast amounts of data. As internal network environments may consist of many thousand hosts (especially in large network infrastructures), it is worthwhile to take the complexity of the used clustering method into account.

Therefore, the following requirements are derived:

R1: Meaningfulness The clustering method should be meaningful, i.e. the clustered hosts should be similar in some way and such that it contributes to finding anomalies;

R2: Stability The clustering should be stable, i.e. clustering the same network under slightly different circumstances should not result in a completely different clustering;

R3: Determinism The clustering should be deterministic, i.e. clustering the exact same network twice should result in the same clustering;

R4: Efficiency The clustering should be efficient, i.e. it should scale to large networks, which can be shown by proving the used clustering has an acceptable complexity.

4.2.3 Evaluation

The next step is to verify the requirements set against the studied clustering methods. First, we describe the used methodology, followed by an overview of the used implementations of the algorithm, concluded by the results per implementation.

4.2.3.1 Methodology

Table 4.11 provides an overview of how it is verified whether a clustering method satisfies our requirements.

Table 4.11: An overview of the used methodology per clustering method requirement.

Requirement	Methodology
R1	Evaluation of type of clustering
R2	Compare clustering changes caused by minor changes in data set
R3	Run clustering on same network number of times, evaluate differences
R4	Evaluation of computational complexity

For the first requirement, it will be argued whether its approach is useful for the intended purpose - answering whether the clustering method is in fact useful for anomaly detection will be determined in the next chapter. The second requirement will be tested by performing a volatility assessment. Testing for determinism, the third requirement, will be achieved by running an analysis on repeatedly applying the same clustering on the same network. If the clustering methods meet the final requirement will be determined by establishing and comparing the computational complexity of the clustering methods.

4.2.3.2 Results

R1 Meaningfulness
k-means and mean shift rely on vector quantisation, as was mentioned before. This implies that in order to cluster a network of hosts using either k-means or mean shift, it is necessary to create a vector of numeric values for each host.

In the context of clustering based on network traffic, it is possible to take host characteristics such as average packet size, average flow size and average flow duration as vector values. This means that the resulting clustering will contain clusters of which hosts have similar behaviour on the selected features. This may prove to be useful in the anomaly detection phase, when it is tested whether hosts behave anomalous by analysing such features. However, since numeric values are required, some properties cannot be taken into account, such as sets (e.g. port numbers used, connected neighbours, up times) and ccategorical features (e.g. an IP address, a port number).

Louvain clustering and Stochastic Blockmodel clustering, on the other hand, are graph-based: these methods do not rely on vectors but on their relation with other nodes, i.e. the edges between the nodes and their respective weight. When applied to network traffic, a strong advantage of graph-based clustering algorithms is that the communication partners of a host are taken into account. As has been analysed before, the communication partners of a host are a useful indicator for identifying malicious behaviour. Clustering on this feature might therefore give meaningful clusters. However, a downside is that other properties such as packet sizes, flow sizes and times active are not taken into account.

Although both relying on graph structures, Louvain and SBM clustering have more profound differences than the other two clustering methods. Louvain clustering looks for communities, i.e. hosts that are close-knit, mainly communicating with each other. Stochastic Blockmodel clustering on the other hand looks at similar connection patterns, i.e. hosts that are similar in the way they interact with other hosts. When considering computer networks, the latter means that SBM clustering is more likely to distinguish the various host types, such as workstations and servers. Intuitively, such clusters might give a better partitioning of similar hosts and are therefore likely to successfully identify harmful anomalies.

In conclusion, all four methods have aspects that contribute to obtaining clusters of hosts that are in some respect similar, and therefore meaningful. Since Stochastic Blockmodel clustering is the only one that comes close to identifying a host's role, we expect it is likely to generate the most meaningful clusters. Nevertheless, the anomaly detection performance in the next chapter will have to show which clustering method results in the best detection performance.

R2 **Stability** In order to verify the stability of the clustering method, it is investigated how the various clustering methods handle minor cluster changes. In network environments, host come and go irregularly; it is desirable for our clustering method to have some tolerance in this aspect.

To test the stability under various circumstances, a number of test are listed in table 4.12. Note that for each of the four tests neither a low nor a high value is desirable. Minor changes should result in more or less the same clustering, while it is expected that many changes will result in a different clustering.

The scores are determined as follows: for each method, it is computed how often the test can

Table 4.12: Overview of stability tests for the assessed clustering methods.

	Type	**Description**
C1	Insert new nodes	Introduce new nodes to the largest cluster. The highest number of nodes that can be inserted without changing the clustering determines the score.
C2	Insert new connections	Introduce new (outgoing) connections to a preselected node. The highest number of connections that can be inserted without changing the clustering determines the score.
C3	Remove nodes	Remove nodes with the lowest connection degree. The highest number of nodes that can be removed without changing the clustering determines the score.
C4	Remove connections	Remove connections from nodes in a preselected cluster. The highest number of connections that can be removed without changing the clustering determines the score.

be repeated until the clustering changes. Thus, in the case of test C1, the score is determined by the number of nodes that can be inserted without changing the clustering. This number is then divided by the number of hosts in the original network, giving the relative change allowed without cluster changes.

The results are given in table 4.13. The low scores for SBM are likely to be impacted by its non-determinism, as this also affects the number of changes without cluster change. *k*-mean shows for the first three test a reasonable number of alterations without cluster change, although removing edges showed to impact the clustering sooner. Mean shift allows the most number of changes for three of the tests. Although this shows that its behaviour is non-volatile, it could also be argued that it is too rigid. Louvain clustering appears to in between the other clustering methods.

Table 4.13: Average number of alterations before cluster changes occur relative to the number of hosts/edges, for the UNB ISCX *2012 data set.*

	***k*-means**	**Mean shift**	**Louvain**	**SBM**
Test C1	3.25%	26.5%	5.75%	1%
Test C2	2%	2.5%	21.75%	1.25%
Test C3	4.25%	100%	3.75%	0%
Test C4	1%	25.75%	0.5%	1%

≥ 99	Too rigid
≥ 95	Rigid
🟨	In between
≤ 5	Volatile
≤ 1	Too volatile

R3 **Determinism** Besides the fact that most clustering algorithms have non-deterministic aspects in their algorithm, the outcome does not necessarily has to be non-deterministic. To determine the extend to which a clustering method's result is non-deterministic, a simple test is executed in which the same clustering method is applied to the exact same data twice. Using the Adjusted Rand Index (see section 4.1.6), the similarity between the two clusterings is determined. By repeating this experiment a number of times, the average ARI scores provides an indication of the the level of determinism. If the average is 1, the clustering was the same each iteration and therefore is a strong indication of determinism (the more iterations, the smaller the probability the clustering is non-deterministic). If the average ARI score is smaller than 1, the clustering method is non-deterministic. A score close to 1 indicates the clustering is more deterministic, whereas a score further from 1 indicates the clustering is rather non-deterministic.

A representation of this test is given as pseudo-code in listing 4.14.

```
function test_R3(data, repetitions) {
    ari = new int[repetitions];
    for(i=0, i<repetitions, i++){
        clustering_1 = cluster_method(data);
        clustering_2 = cluster_method(data);
        ari[i] = compute_ari(clustering_1, clustering_2);
    }
    return min(ari), max(ari), avg(ari);
}
```

Listing 4.14: Pseudo-code representation of test C3.

By determining the ARI of the same networks 1,000 times, at least a 95% percent confidence interval can be given [9]. The results of this test can be found in table 4.15.

Table 4.15: Overview of the minimum, maximum and average ARI *between two exact same network instances for each clustering method, based on 1,000 runs (giving an error margin of* $\alpha = 0.05$*).*

	k-means	**Mean shift**	**Louvain**	**SBM**
MIN	0.99872	1.00000	1.00000	0.35283
MAX	1.00000	1.00000	1.00000	1.00000
AVG	0.99993	1.00000	1.00000	0.75144
Verdict	*(Slightly) Non-deterministic*	*Deterministic*	*Deterministic*	*Non-deterministic*

In summary, mean shift and Louvain are most reliable as they are deterministic. k-means is slightly non-deterministic, with a small probability of producing a different clustering on the networks it was tested on. Stochastic Blockmodel clustering however has proven to be unreliable in the sense that it is clearly non-deterministic.

R4 Complexity We have the following computational complexity levels of the used algorithms:

Table 4.16: Computational complexity (classification) of k-means, means-shift, Louvain and Stochastic Blockmodel clustering. v is the number of vertices, e the number of edges, k the number of clusters, d the number of feature dimensions.

	k-means	**Mean shift**	**Louvain**	**SBM**
Complexity	$O(vkd)$	$O(dv^2)$	$O(v \log v)$	$O(v \ln^2 v)$
Classification	Linear in v [70]	Quadratic in v [70]	Quasi-linear in v [4]	Quasi-linear in v [55]

We assume that d, the number of dimensions for k-means and mean shift, is negligible small compared to v. Similarly, the number of clusters k is assumed to be negligible compared to v.

Based on this analysis, it becomes apparent that mean shift clustering requires a lot of computational power compared to the others. For larger networks, it could take up significant time to establish a clustering, as the complexity is quadratic in the number of nodes. The best performing algorithm is k-means, which computation time grows linear with the number of nodes.

4.2.4 Conclusion

In this second part of the chapter, the applicability of four clustering methods to network traffic data was reviewed based on four requirements. A qualitative overview of the results per requirement is given in table 4.17.

In summary, section 4.2.3.2 has shown that there are severe differences between the investigated clustering methods. Not all clustering methods can fully satisfy the requirements set in section 4.2.2. *k*-means and Louvain clustering seem to be good candidates, while Stochastic Blockmodel clustering might prove to be too volatile to give reliable results and mean shift may result in a bottleneck at the model creation stage.

The next question to answer is what exact impact of these observations on the actual anomaly detection performance of the clustering methods is, as not all requirements might have the same effect on the detection rates.

Table 4.17: Qualitative evaluation of the investigated clustering methods, based on the results of section 4.2.3.

	***k*-means**	**Mean shift**	**Louvain**	**SBM**
R1: Meaningfulness	+	+	+	++
R2: Stability	-	-	·	- -
R3: Determinism	·	++	++	- -
R4: Complexity	+	- -	·	·

++	Excellent
+	Good
·	Fair
-	Bad
- -	Terrible

5 Anomaly detection

In this chapter, the aspects of the anomaly detection to be applied are investigated. Section 5.1 introduces the concept of model-based anomaly detection. Subsequently, section 5.2 discusses which features should be taken into account when applying this type of anomaly detection. Finally, in section 5.3 it is discussed how the features are used to detect anomalies.

5.1 Model-based anomaly detection

5.1.1 Process

Model-based anomaly detection is, as the name suggest, a type of anomaly detection in which models are used to identify anomalous behaviour[8]. Inherent to model-based detection is that prior to using the detection, a model has to be generated based on training data. This model is used for comparison to new data, in which is tested if the new data behaves as determined in the model. Therefore, model-based detection consists of a *training* phase and a *test* phase.

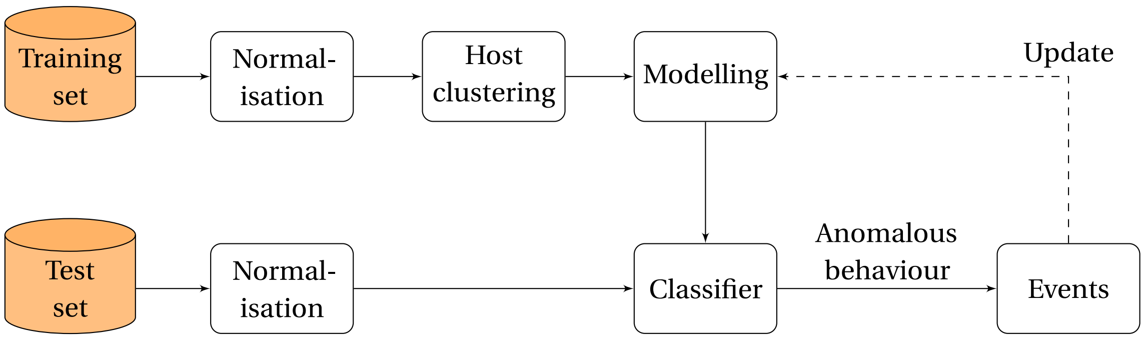


Figure 5.1: Flow diagram of the proposed host clustering-based anomaly detection algorithm.

Training phase Training data is the data used to create the model, which will serve as ground truth. The behaviour of this data is assumed to be the 'normal behaviour' of the network under investigation.

The data is, after gathering, first normalised, which involves parsing the data into a common data format and removing any noise. This is necessary, as the following steps assume that the training data is clean[9] and is a reliable representation of how the concerned

[8] See also chapter 2.2.

[9] 'Clean' in this context means that it does not contain malicious entries, or, in some instances, that the number of malicious entries is a proportion that can be quantified.

hosts behave. Noisy data, which is often caused by misconfigurations, contains entries that are meaningless in the context of detecting malicious behaviour and will pollute the results. Noise should either be resolved at the source or manually removed from the data captures.

After the normalisation, it is possible to obtain a clustering of the network by applying a clustering method. The result of this stage is a mapping for each IP address to a single cluster.

This clustering is used as input for the modelling stage, in which for each cluster is defined what common behaviour is. This definition consists of several *features* that describe certain aspects of the cluster's members' behaviour. Section 5.2 discusses which features are required for the proposed anomaly detection.

Test phase Test data[10] refers to the data being tested, i.e. the data in which the algorithm will look for anomalous behaviour. It should be captured in similar circumstances as the training set, e.g. using the same time frame. The test data is normalised in the same way the training data is.

In this phase, the actual detection takes place. The classifier takes the model derived at the training phase and the normalised data from the test phase and compares them. It is assessed whether the characteristics of the network traffic as observed in the test data matches the behaviour as defined in the model. In other words, for each host, it is tested whether it adheres to the behaviour as defined the model corresponding to the cluster it belongs to.

If behaviour of a host is anomalous to such an extend that it exceeds a set threshold, the behaviour will be marked as anomalous. After the classifier has completed its work, it will return a set of events that should be investigated further.

The resulting events will be evaluated by the operator of the proposed detection system, upon which the operator might decide to take further action. These evaluations may be used for additional training of the model. The advantage of this is twofold: the system gets new, labelled input that will increase its ability to distinguish between anomalous and non-anomalous behaviour, and secondly it allows the system to learn and adapt to changing behaviour.

5.1.2 Data

When it comes to identifying anomalous behaviour in internal network environments, a view on the interactions within the environment need to be provided to the detection mechanism. Since there are multiple types and formats in which network data can be processed, this section will explore which data types and formats are available and which is most suitable for the purposes of this research.

There are various methods and formats in which data can be collected, stored and processed. Especially for model-based anomaly detection, the way in which this is done affects the performance of the detection mechanism. To make clear what the main differences are, consider the following three data sources/types:

[10] In other literature sometimes referred to as *validation data*.

(Full) packet data Packet Capture (or PCAP for short) is an interface for capturing network traffic. PCAP files (PCAPs) form a complete record of network activity, meaning that all data passing the capture interface can be included. Depending on the capturing configuration, network data from all levels from the OSI model, apart of the physical layer, can be collected.

PCAPs are rarely captured for a complete network environment. This is mainly due to the fact that since all network traffic is captured (on packet-basis, usually including the contents of packets transferred) significant processing power and storage capacity is required. The large quantity of information is both PCAP its strength and weakness. Additionally, regarding internal network traffic, sensors should be placed on many locations in the network in order to get a complete view, which often requires thorough infrastructural changes.

Flow data In contrast to PCAPs, flow data does not require deep-packet inspection, but merely focuses on packet headers. Statistics such as source and destination IP address and port, protocols, timestamps and transferred bytes are usually included in flow statistics. Central to the approach taken by flow data is that this data is aggregated per flow, in contrast to PCAPs in which data is by default processed per packet [32].

Flow data is usually collected at router/switch level, although flow data can also be collected using standalone probes. While Cisco's NetFlow is a popular flow data format and widely supported by network equipment vendors, IPFIX is a standardised flow format established by the IETF.

Since flow data only entails a number of statistics for each flow, the resulting files are significantly smaller than PCAPs. However, as with PCAPs, it may be challenging to set up an infrastructure that enables one to capture all flows and aggregate all the data to a single host for further analysis.

DNS queries Another option is to use data from an local DNS query resolver as a means to collect network traffic information. In many network environments, each host is assigned a hostname (e.g. a subdomain of the organisation's domain, or a domain name using a non-default domain extension such as `.local` or `.lan`), which are managed by a dedicated local DNS server. If a host wants to communicate with another host, it sends the DNS server a query containing the destination's hostname. The response will be the destination's current local IP address.

Collecting information from local DNS server provides information about which host queried which host at what time. This information indicates which hosts are likely to have had data exchanges. Because DNS servers are central in nature, a strong advantage of using DNS is that collecting the information is rather easy. However, the data that DNS servers offer contains significantly less information than the aforementioned methods. DNS queries are not the same as data flows: it cannot be certain whether the hosts in fact communicated after a query. Also, DNS queries lack information such as the duration of the actual interaction, the number of bytes transferred, and port numbers used.

The relationship between the data sources can be found in figure 5.2.

The disadvantage of using PCAPs for our purposes is mainly the hard obtainability. Also, it is debatable whether the higher level of information justifies the extra processing power/storage capacity and infrastructural changes required to deploy such a solution. Although flow data only

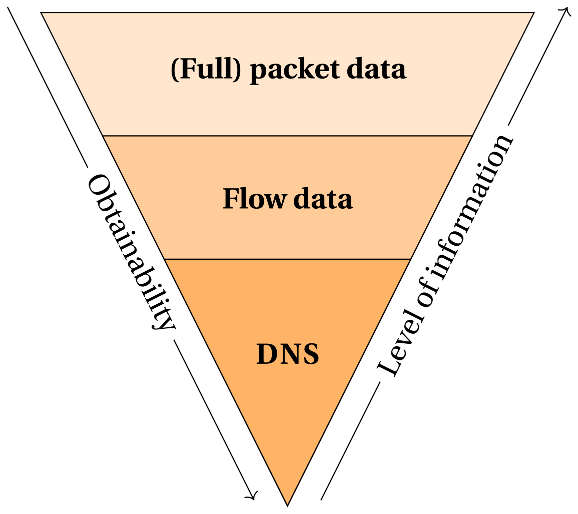


Figure 5.2: Relationship between the level of information and the obtainability for the various data types.

provides a subset of the information PCAPs offer, it is significantly easier to collect. Also, the level of information is sufficient to make statements about interaction characteristics between hosts. Internal DNS queries are probably the easiest to collect, although not every network environment uses internal DNS hostname resolving. Also, as DNS data contains the least amount of data, the possibilities of creating anomaly detection are limited.

Based on these considerations, it is assumed for the remainder of this work that flow data is available as a means to identify anomalies.

Finally, a remark has to be made about the consistency of the data. Regardless of the data type used for anomaly detection, it is important that the data provides a correct view on reality. For instance, in a network environment with DHCP, hosts may get assigned different (internal) IP addresses each time they connect to the network. This could potentially skew anomaly detection results, as an IP address might not reflect the same host over time. For network-based anomaly detection in general, it is therefore assumed that IP addresses are assigned to the same machine over time.

5.1.3 Time frame

Another factor that has impact on the effectiveness of the result is the size for the training data on which the model will be based. In the context of intrusion detection, this comes down to deciding on what the time frame of the training data should be.

The difficulty in this is that the behaviour of user-controlled machines depends on several circumstances: for instance the time of day, day of week, day of month, week, month and year.

If the time frame is too small, it might produce a high number of false positives, as the view on the machine's behaviour might be too narrow. A too large time frame however may make the resulting bounds of 'normal' behaviour too broad, making it hard to distinguish between different circumstances, which may lead to false negatives.

Alternatively, it is possible to use multiple time frames in parallel: for instance, a model that is based on an hours' data, a model that is based on 6 hours' data and model that is based on a day's data. This allows us to observe short-term changes, mid-term changes and longer-term changes in the behaviour of the monitored entities. For the remainder of this work, a time frame of 24 hours is assumed.

5.2 Feature selection

One of the most crucial design decisions of Intrusion Detection Systems is selecting the features. If poorly chosen, the true detection rate may drop, irrespective of the detection algorithm used [50]. For this reason, it will be further investigated which features are relevant for the purpose of intrusion detection through host clustering.

5.2.1 Common approaches

Generally, it is possible to distinguish two methods for deriving a feature set: using data mining or using expert knowledge.

Data mining is one of the most common approaches for feature selection [11, 43, 50]. This requires one or more labelled data sets, which are automatically analysed. In this process, it is analysed which parameters are likely to indicate an intrusion by comparing the network characteristics of intrusive data with those of non-intrusive data. The disadvantage of this approach is that the resulting feature set is likely to successfully identify similar intrusions, but not necessarily other intrusions (as they might impact other network parameters).

The alternative is to select features by using expert knowledge to reason about the selection process [27]. This comes down to 'manual' data mining: by analysing characteristics of intrusions, it is possible to deduct which network parameters are most likely to change. The disadvantage of this approach is that relevant features may be missed out. Additionally, this process tends to be non-deterministic, in the sense that other analysts may end up with a different feature set.

For the purposes of this research, it will not be possible to use data mining to determine the feature set. Due to the lack of data of APT-infected network environments[11], this approach is not suitable for our purposes. Instead, expert knowledge and reasoning will be used. The characteristics of APTs as derived in section 3.3 will serve as resource for picking relevant features.

In the next section, the available data and the relation with the APT characteristics will be discussed. Based on this, a feature set will be determined.

5.2.2 APT-relevant features

Considering the IPFIX format[12], there is a wide variety of statistics per traffic flow available. As not all features may be relevant when it comes to detecting intrusions, it is possible to establish a pre-selection of relevant fields.

In literature, there are a number of recurring features that ought to be used for network anomaly detection. In detecting intrusion attacks, often source/destination IP address, source/destination

[11] This issue is further discussed in chapter 6.1.
[12] See section 5.1.2.

port number, packet size, flow start time and flow duration are taken into account [40]. An overview of these features is given in table 5.3.

Table 5.3: Overview of flow-based features relevant for network intrusion detection, adapted from [40].

Feature	**Type**	**Useful in identifying...**
Source IP address / port	String (Categorical)	• The IP address of infected machines • Failures that might disconnect certain servers or hosts from a network.
Destination IP address	String (Categorical)	• Victims of the infected machine, through e.g. vulnerability probing, port scanning, DoS-ing, etc.
Destination port number	String (Categorical)	• Targeted attacks to specific ports. • Port scanning attacks searching for possible vulnerabilities.
Flow size	Integer (Numeric)	• Increase in packets of certain size, for example increase of 40-byte packets during SYN flooding attack. • Increase in bytes per flow, may indicate *Exfiltrate data* stage. • Increase in packets per session, may indicate C&C channel behaviour.
Flow duration	Float (Numeric)	• Changes in flow duration patterns, e.g., unusual presence of flows of similar durations (scans, DoS attack), server overloads may cause an increase.
Flow start/end	Float (Numeric)	• Changes in times a machine is active, e.g. unusual up or down times of a maching, which might indicate unauthorised access.

Translating this to IPFIX fields, without loss of information, the fields in table 5.4 are needed.

Table 5.4: Overview of relevant IPFIX *fields for network traffic anomaly detection, based on [14].*

Name	**ID**	**Description**
`octetDeltaCount`	1	The number of octets since the previous report (if any) in incoming packets for this Flow.
`packetDeltaCount`	2	The number of incoming packets since the previous report (if any) for this Flow.
`deltaFlowCount`	3	The conservative count of Original Flows contributing to this Aggregated Flow.
`sourceTransportPort`	7	The source port identifier in the transport header.
`sourceIPv4Address`	8	The IPv4 source address in the IP packet header.
`destinationTransportPort`	11	The destination port identifier in the transport header.
`destinationIPv4Address`	12	The IPv4 destination address in the IP packet header.
`flowEndSysUpTime`	21	The relative timestamp of the last packet of this Flow.
`flowStartSysUpTime`	22	The relative timestamp of the first packet of this Flow.
`postOctetDeltaCount`	23	The number of octets since the previous report (if any) in outgoing packets for this Flow.
`postPacketDeltaCount`	24	The number of outgoing packets since the previous report (if any) for this Flow.
`sourceIPv6Address`	27	The IPv6 source address in the IP packet header.
`destinationIPv6Address`	28	The IPv6 destination address in the IP packet header.
`ipVersion`	60	The IP version field in the IP packet header.

In order to show the impact of APTs on the network parameters mentioned, an experiment was conducted on the correlation between attack characteristics and network parameters. For this experiment, a set of clean network traffic data is used to determine the base values for a list of network parameters. Subsequently, each attack characteristic as described in chapter 3.3 is simulated in the data set, thus, resulting in another 6 data sets. Now, by comparing the network parameters of the clean data set with the infected data sets, it is possible to make statements about the impact of APT characteristics on internal network traffic parameters. The results of this experiment can be found in table 5.5.

For each attack characteristic, various attack scenarios where tried. Using a scoring system, an average of the impact of the various attacks is provided. As APTs are sophisticated, unique and change over time, the list of possible scenarios is non-exhaustive. The table should therefore be seen as a rough estimation of the relation between the various attack characteristics and internal network features.

The impact is expressed as a three-level qualitative measure. A red indicator means that in most of the scenarios, the respective feature was significantly changed. The yellow indicator means that in some cases, but not all, this corresponding feature was affected. The grey indicator implies the corresponding feature was not affected or only slightly changed.

To compute score s_i^k of a characteristic i for feature k, we use:

$$s_i^k = \frac{\sum_{j \in J} s_{ij}^k}{|J|} \text{ where } s_{ij}^k = \begin{cases} \dfrac{x_{new} - x_{old}}{x_{old}} & \text{numerical feature} \\ \dfrac{|x_{new} \cup x_{old}| - |x_{new} \cap x_{old}|}{|x_{old}|} & \text{categorical feature} \end{cases} \tag{5.1}$$

where x_{new} is the value of the characteristic with the attack (for feature k of characteristic i in scenario $j \in J$), while x_{old} is the value in a situation without attack.

For $|1 - s_i| > 0.05$ the indicator is red; for $0.05 \leq |1 - s_i| < 0.005$ the indicator is yellow; for $|1 - s_i| \leq 0.005$ the indicator is grey.

Table 5.5: *Likelihood of observing behaviour in internal network traffic for the analysed* APT *campaigns, for each of the characteristics from section 3.3.*

		EA1	EA2	SF1	SF2	SF3	PA1	PA2
Hosts	New hosts appear	🟥	⬜	⬜	⬜	⬜	⬜	⬜
	Hosts disappear	⬜	⬜	⬜	⬜	⬜	⬜	🟨
Partners	New partners for cluster	🟥	⬜	🟨	🟨	🟨	🟨	🟨
	No longer partners for cluster	⬜	⬜	⬜	⬜	🟥	⬜	⬜
Ports	New ports for partners for cluster	🟥	🟥	🟥	🟥	🟥	🟨	🟨
	No longer port for partner for cluster	⬜	⬜	⬜	⬜	🟥	⬜	⬜
Traffic	Bytes/packet	⬜	⬜	⬜	⬜	⬜	🟨	⬜
	Bytes/session	⬜	⬜	⬜	🟨	🟨	🟥	🟨
	Bytes/time unit	🟨	🟨	🟥	🟨	🟥	🟥	🟨
	Packets/session	🟨	🟨	🟨	🟨	🟨	🟥	🟨
	Packets/time unit	🟥	🟥	🟥	🟨	🟥	🟥	🟨
	Duration	⬜	⬜	🟨	⬜	🟥	🟨	⬜
Time	Bytes sent over time	⬜	⬜	🟥	🟨	🟥	🟥	⬜
	Packets sent over time	🟨	🟨	🟥	🟨	🟥	🟥	🟨
	Sessions over time	🟨	🟨	🟥	🟨	🟥	🟥	🟨

Legenda: ⬜ No impact 🟨 Small impact 🟥 Significant impact

5.3 Anomalous behaviour

After having established the set of features to take into account when looking for anomalies, the definition of anomalous behaviour should be formulated. In other words, the next question to answer is based on what criteria a host (or a flow) should be marked as anomalous.

The answer to this question highly depends on the classification algorithm chosen to process the data. As is shown in figure 5.1, the classifier is the core part of the anomaly detection algorithm, as it compares the test data against the training data and assigns anomaly scores to the test data. This process of comparing data and assigning scores can be implemented in several ways. Classification algorithms applied in existing anomaly detection approaches include k-nearest neighbour, Support Vector Machine, Local Outlier Factor, Decision Tree and Bayesian Networks [18].

Even though the chosen classification algorithm has a significant impact on the overall result of the anomaly detection technique, it is not possible to compare all available classification methods after the proposed clustering step. As choosing the best classification algorithm is a complete science in itself, it is considered to be outside the scope of this work. Instead, the performance improvements on a single classification algorithm will be investigated.

Hence, to illustrate the performance improvement caused by the proposed clustering step, a Support Vector Machine classifier will be used, a classifier that has proven to have a good performance when applied to network traffic data [12, 22].

5.3.1 Support Vector Machines

Support vector machines (SVMs), proposed by Boser et al. [6], estimate one or more regions in a feature space in which most of the data occurs. This region can be used to predict if a new data instance is similar to the training set. Although the standard Support Vector Machine algorithm requires a labelled data set, Schölkopf et al. [63] present an adapted version in which unsupervised learning is supported. In general, the advantage of SVM for anomaly detection is that despite the relatively high training time, they are considered to provide high accuracy [30].

The algorithm of SVM uses a nonlinear mapping to transform the training data into higher dimensions. Using the density of the points, the linear optimal separating hyperplane is computed, which is the decision boundary for separating points from one class from another. Given an appropriate nonlinear mapping to a sufficiently high dimension, it is always possible to separate two classes of data using a hyperplane (given the data consists of at least two unique points). This hyperplane is found using support vectors and margins, and acts as decision surface, as will be further explained in section 5.3.2.

Depending on the type of data being analysed, one might expect multiple classes of data, resulting in multiple hyperplanes. One-Class SVM is an instance of SVM in which a single class of data is sought after, and hence only has a single hyperplane. In the context of anomaly detection, in which a distinction between normal and anomalous behaviour should be made, this often suffices, the single class being normal data instances.

Given a training data set of n points, unsupervised SVM tries to separate the entire set of training data from the origin, with the underlying assumption that the training data is clean. This separation is achieved by solving a quadratic program that penalises any points not separated from the origin while simultaneously trying to maximise the hyperplane's distance from the origin.

More specifically, the optimisation comes down to

$$\min_{w\in Y,\xi_i\in\mathbb{R},\rho\in\mathbb{R}} \left[\frac{1}{n}\sum_{i=1}^{n}\max\left(\xi_i-\rho\right)\right]+\frac{1}{2}\|w\|^2 \qquad (5.2)$$
$$\text{subject to } (w\cdot\phi(\vec{x}_i)) \geq (\rho-\xi_i), \xi_i \geq 0$$

for feature set w, data instance $\vec{x}_i$, the number of data instances n, offset from the origin ρ, and transformation function $\phi(x)$. Variables ξ_i are slack variables, which penalise the objective function while allowing some points to be on the wrong side of the hyperplane.

Although the most basic version of SVM works with a linear separator, using a nonlinear kernel is likely to give more accurate results as it cannot perform worse than linear separators [39]. The kernel function $K(x,\cdot)$ defines the distribution of similarities of points around a given point x, i.e. $K(x,y)$ denotes the similarity of point x with point y. SVM is often used together with

Gaussian RBF $K(\vec{x}_i, \vec{x}_j) = e^{-\gamma\|\vec{x}_i - \vec{x}_j\|^2}$ having $\gamma = \frac{1}{2\sigma^2}$ by default. Alternatively, SVM can be used along with the Sigmoid or the Polynomial kernel [30]. Transformation function $\varphi(\vec{x}_i)$ should satisfy $K(\vec{x}_i, \vec{x}_j) = \varphi(\vec{x}_i) \cdot \varphi(\vec{x}_j)$.

The computational complexity of standard SVM is $O(n^3)$ for training for the n the number of training samples. Although this cubic complexity means SVM training scales worse for large data sets, training usually is a one-off event at the beginning of deploying an SVM-based anomaly detection mechanism. Additionally, as was argued in section 5.1.3, the number of input samples may be limited, meaning the actual computational burden might turn out to be low. For SVM prediction (i.e. the test phase), the complexity for most SVM kernels is $O(n_{SV}d)$ for d input dimensions and n_{SV} the number of support factors. An approximation exists for RBF kernels to have a computational complexity of $O(d^2)$ [13]. As the number of input dimensions is usually a low, fixed number, this means that using SVM near or in real-time is a feasible option.

As an anomaly classifier, SVM is applied in multiple network traffic anomaly detection approaches. Eskin et al. [22] apply One-Class SVM to all flow data provided by the KDD CUP'99 data set[13]. Their research shows that SVM is able to identify anomalous traffic flows with a 98% true positive rate and a 10% false negative rate, outperforming the *k-nearest neighbour* and *cluster-based estimation* classifiers. Li et al. [44] show a modification of SVM using a linear kernel function which the authors claim makes the algorithm more lightweight in processing data. This optimisation is especially useful for applying SVM as anomaly classifier in large network environments, where it is supposed to handle vast amounts of data.

5.3.2 Applying data and scoring

Each point has a distance from the closest hyperplane: the distance is positive if it is outside a hyperplane, while it is negative if it is inside the hyperplane. It is possible to set a threshold d_{thres}, which indicates the maximum distance a data instance may have to be considered as non-anomalous. All data instances having a distance larger than d_{thres} are considered to be anomalous. By default, $d_{thres} = 0$.

While SVM works with numerical features, it is possible to add categorical features. For adding a categorical feature X to feature space w, a column for each possible value of X is added to w. Each data instance sets the relevant column to 1 if it is the value of X, while it sets the other columns to 0. This ensures the projected distance between two instances with a different value for X is constant. An example to illustrate this principle is provided in figure 5.6.

Because SVM is kernel-based and the used features will have different units (bytes, seconds, etc.), it is necessary to normalise the data in order to make different dimensions comparable. Without this normalisation step, changes on features with a large domain have more impact on the distance than changes on features with a smaller domain. To overcome this, the raw data is normalised in order to obtain a data-dependent normalisation feature map. As a result, the feature maps corrects each continuous value as the number of standard deviations it is away from the mean. In other words,

$$\vec{x}\,' = \frac{\vec{x} - \mu_x}{\sigma_x} \tag{5.3}$$

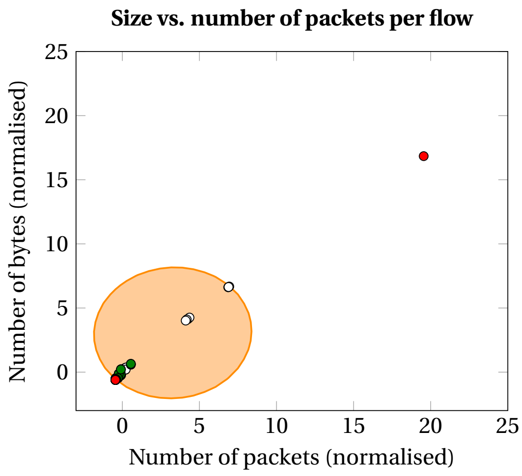
for feature vector $\vec{x}$, average μ_x and standard deviation σ_x for feature x.

[13]This data set is examined in chapter 6.1.

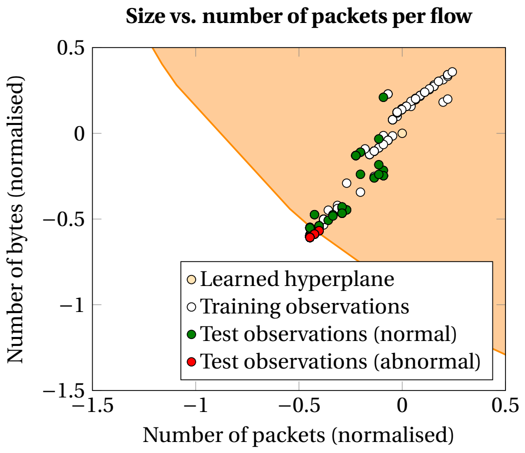
$$w = \begin{bmatrix} f_1 & f_2 & f_3 & f_4 & \dots \\ 123 & 12 & 45.6 & B & \dots \\ 212 & 48 & 22.0 & A & \dots \\ 5 & 22 & 42.1 & B & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \longrightarrow w' = \begin{bmatrix} f_1 & f_2 & f_3 & f_4 = \mathrm{A} & f_4 = \mathrm{B} & \dots \\ 123 & 12 & 45.6 & 0 & 1 & \dots \\ 212 & 48 & 22.0 & 1 & 0 & \dots \\ 5 & 22 & 42.1 & 0 & 1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Figure 5.6: Example of adding a categorical feature f_4 to numeric feature set w.

A simplified example of SVM is given in figure 5.7 with a (normalised) two-dimensional feature space consisting of number of bytes versus number of packets per flow. By setting d_{thres} to 0 (i.e. all test observations outside the hyperplane are marked as anomalous), this basic SVM has a false positive rate of 43% with a false negative rate of 4.4%. The detection rates increase significantly by adding more dimensions.



(a) Overview



(b) Zoom with $x, y \in [-1.5, 0.5]$.

Figure 5.7: Example of a two-dimensional SVM *with training data and test data using the* RBF *kernel with $\gamma = 0.001$.*

6 Evaluation

Given our proposed approach in chapter 5, this chapter explores whether the new approach is an improvement compared to the *status quo*. To this end, section 6.1 investigates the available data sets that can be used for this purpose. Section 6.2 introduces the proof-of-concept used for the analysis followed by section 6.3 presenting the results of the evaluation for the preprocessing, clustering and classification steps, respectively.

6.1 External data sources

In order to be able to evaluate the proposed methodology, external data sets will be evaluated using our approach.

A common issue in the research domain of intrusion detection is the lack data available for the evaluation/verification of new methods. In the context of this research, it would have been preferable to have access to multiple enterprise-scale network environments, having full captures of all network traffic, which are attacked by APTs. Unfortunately, data sets of network traffic caused by APTs that are available for research purposes are scarce, for multiple reasons. The foremost reason is that APT-like attacks often target large organisations, which are reticent in sharing such sensitive information. Additionally, concerns over privacy, confidentiality and new security breaches makes obtaining such sets hard.

There are, however, some network data sets available in the public domain, which can be used for scientific purposes. In literature, there exist several overviews of available data sets for evaluating IDS-related solutions [64, 68]. These overviews are often accompanied with reviews of the data sets, indicating how useful the reviewed sets are for specific purposes. None of the overviews addresses aspects that are particularly releavant to our research, such as the availability of internal network traffic and the size of the internal network. For this reason, a set of requirements in context of this research is defined, followed by an evaluation of the data sets mentioned in the data set overview literature.

6.1.1 Requirements

The requirements for using our approach can be split into two categories: requirements for using the method and requirements for evaluating the method. The latter is an extension of the former, as properly evaluating our approach leads to some more preconditions.

6.1.1.1 Use of the proposed method

First, one can define the requirements for using the algorithm:

The data set should contain internal network traffic. As this research focuses on changes in internal network traffic patterns, it is necessary that the data set contains this type of traffic. Data sets exclusively consisting of network traffic between internal hosts and external hosts (i.e. Internet traffic) are useless in this context.

The data set should comprise multiple captures of the same network. Due to the nature of the proposed anomaly detection mechanism, both training data and test data are required. This implies that at least two, preferably more, captures of the same network should be present in order create a model of the network and use this to test for anomalies. Hence, a single capture of a network is useless in this context.

Any data set not satisfying these requirements is unsuitable for use of the proposed method. Data sets that do satisfy these requirements, however, can be used to derive a model and compare the model against the remainder of the data.

6.1.1.2 Evaluation of the proposed method

When it comes to the evaluation of the anomaly detection method, the data set should satisfy the following additional requirements:

The data set should contain one or more internal network attacks. In order to see whether the algorithm successfully identifies anomalies caused by intrusions, it is necessary to have captures of both normal and attack situations. Conversely, it is also used to determine whether the algorithm does not erroneously marks a normal event as anomalous.

The data set should be labelled. Without labelling, evaluating the performance of new intrusion detection methods is hard, as it would require manual interpretation of the data set. While it may be feasible to say something about false positives (by applying the algorithm and subsequently analysing the events marked as anomalous), it is practically impossible to make statements about false negatives, as this would require an analysis of the complete data set. For this reason, having a labelled data set is vital for the evaluation of the detection performance.

The data set should contain recent data. As network intrusions are evolving, it is important that the data set contains traffic that reflects today's attack practises. Similarly, network environments have evolved as well: network infrastructures have become increasingly complex, as for instance mobile devices are commonplace. For convenience, we quantify this condition by requiring the set to have been captured within the last 5 years.

The data set should reflect a realistic network environment. The internal network traffic should reflect common network infrastructures as seen in corporate networks. This involves client-server infrastructures consisting of numerous hosts. This makes it possible to make statements about the applicability to similar networks.

Using sets fulfilling these requirements gives us the opportunity to make reliable statements about our approach.

6.1.2 Evaluation

Based on the overviews provided by Shiravi et al. [64] and Torrano Giménez et al. [68], the available data sets are assessed for each of the aforementioned requirements. An overview of the results are given in table 6.1. In the next sections, the individual data sets are discussed.

Table 6.1: *Overview of publicly available network data sets.*

	Usage		Evaluation				
	Number of active, internal hosts	**Consists of multiple captures**	**Contains normal data**	**Contains attack data**	**Labelled data**	**Age of data (years)**	**Realistic network configuration**
Caida 2008-2016 [10]	0	Yes	Yes	No	No	≤ 1	Yes
DEFCON 2009-2015 [19]	Varying	Yes	No	Yes	No	≤ 1	No
KDD'99/DARPA'99 [67]	$O(10^1)$	Yes	Yes	Yes	Yes	≥ 15	Yes†
LBNL'05 [51]	$O(10^3)$	Yes	Yes	No	No	≥ 10	Yes*
UNB ISCX'12 [64]	$O(10^1)$	Yes	Yes	Yes	Yes	≤ 5	Yes†

6.1.3 Caida 2008-2016

These sets are offered by the Center of Applied Internet Data Analytics, part of the University of California (San Diego, U.S.A.) [10]. The traces in the set are collected from two monitors on a commercial backbone link since April 2008. Since no internal network traffic is captured, the set does not qualify for use in this research.

6.1.4 DEF CON 2009-2015

The *DEF CON* Capture the Flag (CTF) data sets are part of the annual *DEF CON* conference [19]. The data sets are generated during the competition, meaning the data sets mainly contain intrusive traffic. There are multiple captures available captured over the last years. However, the *DEF CON* data sets do not reflect a realistic network configuration nor do they contain 'normal' network configuration, as a result of the setting in which they are captured. For this reason, they are unsuitable for our evaluation purposes.

6.1.5 KDD 1999 and DARPA 1999

The *KDD* and *DARPA* data sets [67], both generated in the year prior to the 21th century by the Massachusetts Institute of Technology (MIT), have been frequently used in literature. It is a simulated data set based on network traffic seen in a medium-sized US Air Force base. The data set comprises multiple captures with both normal traffic and attack traffic. Additionally, the set is supplied with labelling.

The reason why this data set is nevertheless unsuitable for evaluation is the fact that the data is over fifteen years old. The network patterns do not represent current trends, neither do the attacks reflect realistic, modern threats. Using this data set for evaluation would give an unrealistic view on network attacks. Additionally, others have found numerous irregularities due to the data generation approach and have criticised the set for failing to resemble real traffic

*This set contains only inter-subnet traffic.

†This set is simulated, based on real network data.

[8]. The *KDD'99* data set is also based on the *DARPA* sets, suffering from the many of *DARPA*'s issues.

6.1.6 LBNL 2005

This set comprises over a hundred hours of data, captured at two internal network locations at the Lawrence Berkely National Laboratory (LBNL), located in the United States of America [51]. The network traffic captured originates from an enterprise network, containing several thousand hosts. This makes this set suitable for usage of the proposed method.

A disadvantage of this data set is the fact that the data is over 10 years old. Also, the set only covers five non-sequential days and although it contains internal network traffic, only inter-subnet traffic is captured. The latter implies that data flows within subnets are not included, meaning this set only provides a limited view on the internal network communications. Additionally, as there is no attack data present, it is not possible to evaluate the algorithm using this set.

6.1.7 UNB ISCX 2012

The UNB ISCX *2012* set [64] is provided by the Information Security Centre of Excellence (ISCX), part of the University of New Brunswick, Canada. The set comprises seven consecutive days (11th of June - 17th of June).

The data included in the set is simulated data based on characteristics of real network data. The authors analysed four weeks worth of network activity of the network environment at the ISCX for this. Using the analysis, the researchers weer able to extract attack profiles based on statistical probabilities. The advantage of this approach is that the full capture (including Application Layer level data) is available without anonymisation. A downside however is that generated data may fail in representing realistic data as important details might have been lost.

For each day of the data set, there are several dozen hosts that participate in internal network traffic. Most interestingly, three of the seven days contain malicious internal network flows. Since most data is labelled (albeit only the last six days), this set provides us everything that is necessary to use this set for the evaluation of our proposed anomaly detection method. In contrast to the *KDD'99* and *DARPA'99* set, this data is more recent, providing a more realistic view on contemporary internal network traffic patterns and attacks.

6.2 Proof-of-concept

In order to evaluate the proposed anomaly detection technique, a proof-of-concept was developed that implements the concept as presented in chapter 5. A more elaborate overview of the proof-of-concept is provided in appendix A.

6.3 Results

The results of the evaluation of the proposed method using the proof-of-concept are discussed in this section.

Using the UNB ISCX 2012 set, the methodology as set out in chapter 5 is carried out. The process is described in three steps: first, preprocessing is are applied to the data in order to prepare the data for the actual analysis. Secondly, the four clustering methods are applied to the data, prior to model creation, resulting in a set of models per clustering method. Finally, the obtained models are used for the classification step in order to identify malicious behaviour. In this step, the detection rates for the different clustering methods are discussed.

6.3.1 Preprocessing

Using the detection algorithm starts with preprocessing the data, which entails filtering out internal network traffic, putting the data in a common data format and removing noise.

Filtering is applied based on the prefix of the source and destination source address: as defined by the authors of the data set, hosts within the network always have an IP address in the `192.168.0.0/16` block. As table 6.2 shows, the UNB ISCX 2012 data contains tens of thousands internal network traffic flows per day. The labelling indicates that only data of Sunday 13 June, Monday 14 June and Tuesday 15 June contain attacks that take place in the internal network.

Table 6.2: Flow-related statistics extracted from the UNB ISCX *2012 data set and its labelling.*

Data set	**N/o flows**		**N/o malicious flows**	**N/o hosts**
	Total	Internal	Internal	Internal
`testbed-12jun`	133,193	27,137	0	3 3
`testbed-13jun`	275,528	58,037	19,539	145
`testbed-14jun`	171,380	25,347	3,624	102
`testbed-15jun`	571,698	74,370	37,230	37
`testbed-16jun`	522,263	50,041	0	36
`testbed-17jun`	353,992	34,397	0	33

The first step is to remove noise. To some degree, this ‘noise’ can be seen as anomalous behaviour. It is arguable that this behaviour is abnormal and unwanted, and hence detecting this counts as a true positive. Yet, as our goal is to identify malicious behaviour from a security perspective, this kind of network patterns could obfuscate the actual patterns that are going on. Even worse, the noise might make it harder to identify truly malicious behaviour. Therefore, upon detecting noise, it should be resolved as much as possible at the source. Since in our case this is impossible, the noise will be stripped from the data if necessary.

By manually analysing the data, the data does not appear to show heavy noise. This was confirmed in the next steps, when clustering and classification did not result in unexpected results. The fact that this data set was simulated and is not a ‘real’ set might explain this observation. Other, non-simulated sets have shown to contain host/application misconfiguration that result in irregular, unusual patterns that skew the results of the detection mechanism.

Next, the relevant data was extracted from the data set and put into a common data format, which for each flow contains: the IP address and port number for both source and destination, the number of incoming packets and bytes, the number of outgoing packets and bytes, the relative start time and duration in seconds, the incoming and outgoing transfer rate in bytes per second, and the label (either `Normal` or `Attack`). As the labelling is provided on flow-basis, the definition of flows as used in this data set are adopted. For TCP traffic, a flow is defined as a TCP connection from establishment to termination, while it is unclear how UDP flows are defined [64].

6.3.2 Clustering

Next, the four investigated clustering methods (k-means, mean shift, Louvain and Stochastic Blockmodel clustering) are separately applied to the given data. For k-means and mean shift, the number of incoming and outgoing connections, the average number of bytes in and out, the average number of packets in and out and the average duration of incoming and outgoing connections are taken as features. For both Louvain and Stochastic Blockmodel clustering the input graph is weighted with the number of connections.

The first observation is that the four clustering methods indeed result in four rather different clusterings. Table 6.3 provides an overview of the number of clusters and their respective cluster sizes for each day in the UNB ISCX 2012 data set.

The table also shows that despite the second and third data set contain 3-5 times more hosts than the others, the number of clusters remains in most cases about the same size. The new hosts are mainly placed into the same cluster, which results in having one relatively big cluster in those days. This stability of the number of hosts is in line with earlier made remark that the number of clusters (and hence models) scales well with an increasing number of hosts.

Table 6.3: The number of clusters and their cluster sizes for each each data entity in the UNB ISCX *2012 data set.*

Data set	**k-means**	**Mean shift**
`testbed-12jun`	8: {11, 11, 4, 2, 2, 1, 1, 1}	6: {15, 11, 4, 1, 1, 1}
`testbed-13jun`	8: {114, 11, 11, 4, 2, 1, 1, 1}	16: {111,8,7,6,2,1,1,1,1,1,1,1,1,1,1,1}
`testbed-14jun`	8: {80, 16, 1, 1, 1, 1, 1, 1}	5: {97, 2, 1, 1, 1}
`testbed-15jun`	8: {18, 9, 3, 3, 1, 1, 1, 1}	3: {35, 1, 1}
`testbed-16jun`	8: {14, 10, 4, 3, 2, 1, 1, 1}	5: {31, 2, 1, 1, 1}
`testbed-17jun`	8: {12, 7, 5, 4, 2, 1, 1, 1}	4: {23, 8, 1, 1}

Data set	**Louvain**	**SBM**
`testbed-12jun`	5: {10, 9, 6, 6, 2}	4: {15, 10, 7, 1}
`testbed-13jun`	3: {72, 67, 6}	3: {111, 32, 2}
`testbed-14jun`	6: {72, 9, 8, 6, 5, 2}	5: {66, 24, 10, 1, 1}
`testbed-15jun`	6: {10, 9, 7, 6, 3, 2}	3: {25, 10, 2}
`testbed-16jun`	5: {12, 8, 8, 6, 2}	4: {17, 11, 7, 1}
`testbed-17jun`	6: {9, 9, 5, 4, 4, 2}	4: {15, 10, 7, 1}

6.3.3 Classification

The next step is to identify malicious flows and hosts based on the models created in the previous section. The goal of this section is to compare the performance of anomaly detection with and without clustering.

For the comparisons to be given, the investigated clustering methods (k-means, mean-shift, Louvain, Stochastic Blockmodel clustering) are compared to applying no clustering at all and per-host clustering. The latter *de facto* comes down to creating a model per host.

In order to do so, the number of false positives is set out against the number of true positives. Generally, the number of false positives (i.e. entities wrongly identified as being malicious) should be a small as possible while the number of true positives (i.e. events correctly identified as being malicious) should be as high as possible. An overview of all rate types are given in table 6.4.

As in most cases a perfect detection score of 0% false positives, 100% true positives (irrespective of the data set used) is very hard to achieve, there is a trade-off between the number of false positives and true positives. Commonly, a higher number of true positives results in a higher number of false positives. Conversely, a lower number of false positives will result in a lower number of true positives.

Each entity (either a flow or a host) is awarded an anomaly score, which is a measure of abnormal behaviour. In evaluating the scores, the central question is what the minimal score an entity should have be before marking it as anomalous. Although this threshold directly impacts the true and false positive rates, there is no universal threshold that gives the best results - it is dependent on the environment the detection mechanism is used in. To overcome this issue, the relationship between true and false positives is often represented in a Receiver Operating Characteristic (ROC). In the ROC, each point represents the number of true and false positives for a given threshold.

Table 6.4: Overview of relationship between possible performance classifications. A means marking an entity as anomalous, I means the entity actually being intrusive.

Probability	**Class**	**Formula**
$\Pr(A \mid I)$	True Positive Rate	$TPR = \frac{\#TP}{\#FP + \#TN}$
$\Pr(A \mid \neg I)$	False Positive Rate (*Type I error*)	$FPR = \frac{\#FP}{\#TP + \#FN}$
$\Pr(\neg A \mid I)$	False Negative Rate (*Type II error*)	$FNR = \frac{\#FN}{\#TP + \#FN}$
$\Pr(\neg A \mid \neg I)$	True Negative Rate	$TNR = \frac{\#TN}{\#FP + \#TN}$

The optimal threshold depends on the environment in which the anomaly detection mechanism is deployed: in high-security environments, a high true positive rate is often vital, while the resulting higher false positive rate is taken for granted. In other environments, lower false positive rates might be more desirable to avoid the burden of processing a vast number of alerts. In most cases however, the threshold for which $TPR + (1 - FPR)$ is maximal provides the best balance between false and true positives. A measure for this is the accuracy ($ACC = \frac{\#TPR + (1 - \#FNR)}{2}$).

Note that for this comparison, we did not update the model after classification, but rather kept the same model. As a result, each test data set is compared against the same training data.

6.3.3.1 Per-flow classification

Using the proposed method as an intrusion detection system, one applies the mechanism on flows. This means that each flow gets assigned an anomaly score, based on the features of that flow. By setting a threshold score for which a score is considered to be anomalous, it is possible to identify anomalous flows.

Given that our data set is labelled, it is possible to generate ROC curves for each clustering, using the threshold score as variable. The ROC curves using the UNB ISCX 2012 data of Saturday 12 June as model for analysing Sunday 13 June, Monday 14 June and Tuesday 15 June are provided in figure 6.5. For each clustering method, the true positive rate and false positive rate are given for the threshold with the highest accuracy in table 6.6.

There are a number of conclusions that can be drawn from these results. First of all, it shows that in most situations clustering performs better in terms of detection compared to using no clustering or per-host clustering. We see that Stochastic Blockmodel clustering in all cases is better than applying no clustering. Both mean-shift and Louvain compare in some cases better, in some cases worse than no clustering. k-means performs in all cases worse than applying no clustering.

Yet, per-host clustering is in all situations clearly the worst performer. Even though it eventually always reaches a high true positive rate, it suffers from high false positive rates. This observation is in line with our hypothesis that per-host clustering is volatile when it comes to minor abnormalities.

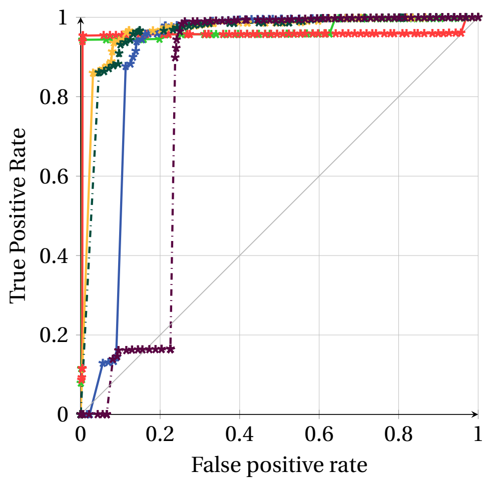
6.3.3.2 Per-host classification

Within the scope of APT detection, it makes more sense to look at the classification of hosts rather than the classification of flows. For instance, an infected machine might initiate flows that are in itself barely anomalous, but the number of these flows together makes the machine's behaviour anomalous altogether.

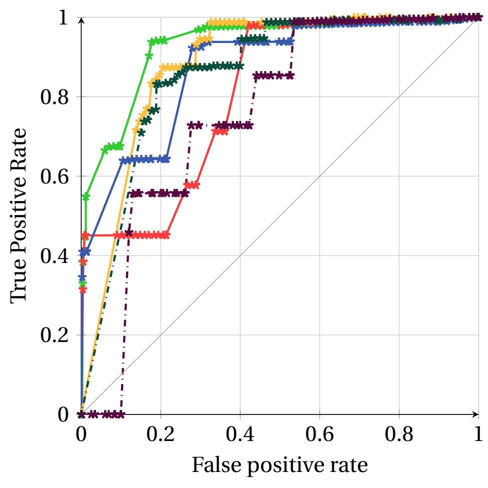
Therefore, it is also useful to generate ROC graphs for the detection rate of anomalous hosts. Figure 6.7 shows the ROC graphs per clustering method for Sunday the 13th of June till Tuesday the 15th of June, using Saturday the 12th of June as model. Table 6.8 provides the true positive rates and false positive rates for the highest accuracy per clustering.

It can be seen that on Sunday all methods are able to give the best result. This is likely due to the fact that the attack this day is clearly visible in the traffic data. For the 14th of June, all clustering methods perform better than both not applying clustering and clustering per host. The same conclusion applies to the 15th of June, although the difference is slightly smaller.

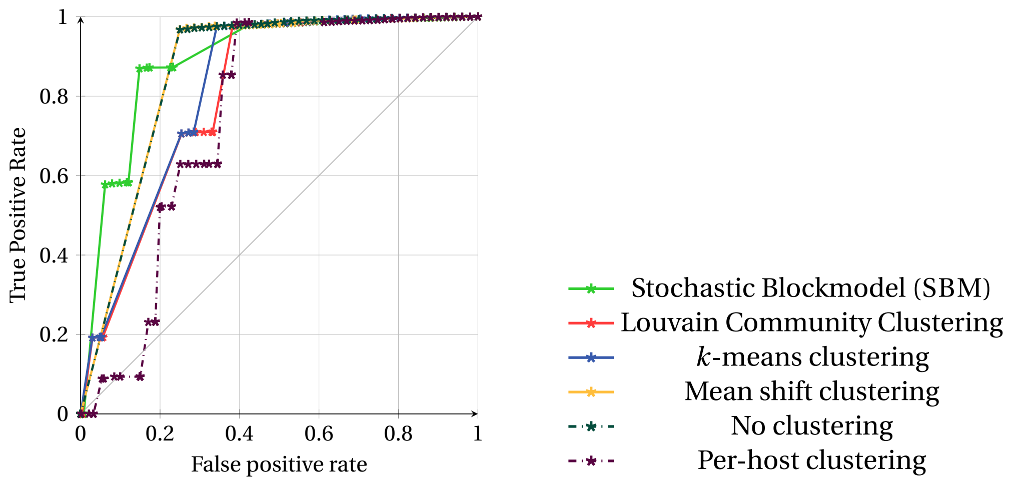
As can also be seen from the graphs, the limited number of hosts in the data result in somewhat sketchy ROC curves; a larger data set would have given more reliable results. Still, the graphs clearly indicate that the performance is never worse (for none of the clustering methods) and in most cases even better.



(a) Sunday 13 June



(b) Monday 14 June

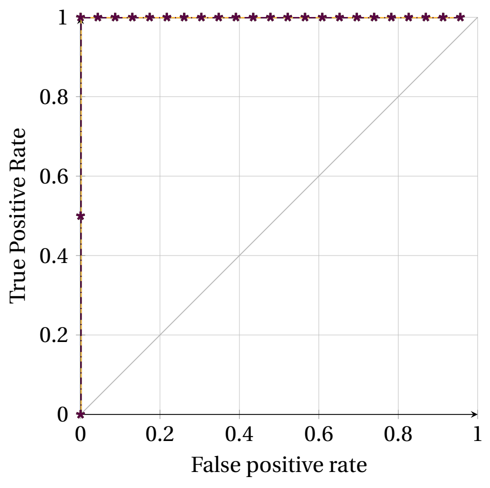


(c) Tuesday 15 June

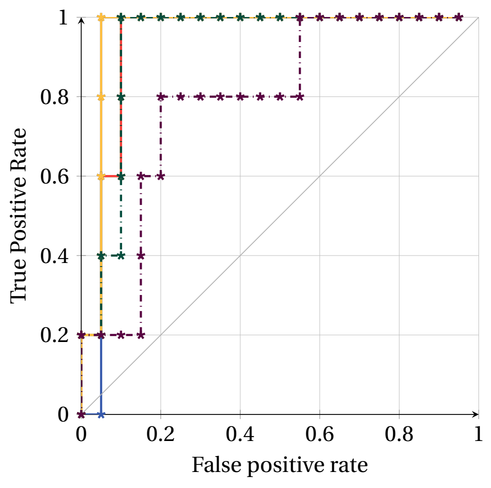
Figure 6.5: ROC *curves for flows based on comparing* UNB ISCX *2012 data from Saturday 12 June against Sunday, Monday and Tuesday, respectively.*

Table 6.6: The true positive rate (TPR)*, false positive rate* (FPR) *and the accuracy* (ACC) *for the highest accuracy for each clustering per data set, based on flows.*

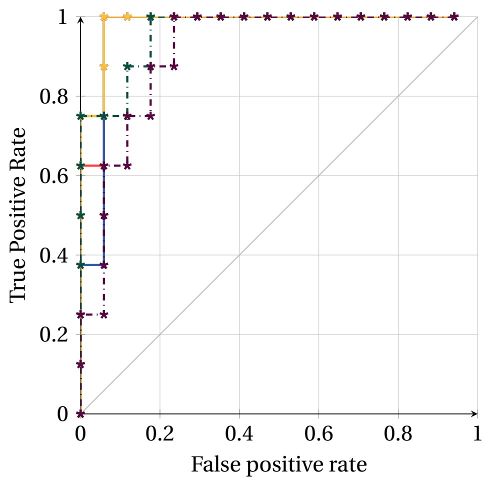
	Sunday			**Monday**			**Tuesday**		
	TPR	FPR	ACC	TPR	FPR	ACC	TPR	FPR	ACC
SBM clustering	0.943	0.005	0.969	0.938	0.177	0.881	0.869	0.148	0.861
Louvain clustering	0.954	0.006	0.974	0.979	0.421	0.779	0.976	0.382	0.797
***k*-means clustering**	0.939	0.141	0.899	0.923	0.279	0.822	0.975	0.343	0.816
Mean shift clustering	0.936	0.083	0.927	0.873	0.206	0.834	0.968	0.250	0.860
No clustering	0.931	0.100	0.916	0.832	0.189	0.821	0.967	0.250	0.859
Per-host clustering	0.984	0.255	0.865	0.989	0.536	0.727	0.985	0.393	0.796



(a) Sunday 13 June



(b) Monday 14 June



(c) Tuesday 15 June

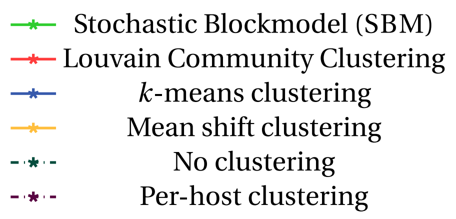


Figure 6.7: ROC *curves for hosts based on comparing* UNB ISCX *2012 data from Saturday 12 June against Sunday, Monday and Tuesday, respectively.*

*Table 6.8: The true positive rate (*TPR*), false positive rate (*FPR*) and the accuracy (*ACC*) for the highest accuracy for each clustering per data set, based on hosts.*

	Sunday			**Monday**			**Tuesday**		
	TPR	FPR	ACC	TPR	FPR	ACC	TPR	FPR	ACC
SBM clustering	1.000	0.000	1.000	1.000	0.100	0.950	1.000	0.059	0.971
Louvain clustering	1.000	0.000	1.000	1.000	0.100	0.950	1.000	0.059	0.971
***k*-means clustering**	1.000	0.000	1.000	1.000	0.050	0.975	1.000	0.059	0.971
Mean shift clustering	1.000	0.000	1.000	1.000	0.050	0.975	1.000	0.059	0.971
No clustering	1.000	0.000	1.000	1.000	0.100	0.950	1.000	0.176	0.912
Per-host clustering	1.000	0.000	1.000	0.800	0.200	0.800	1.000	0.235	0.882

6.4 Conclusion

From the results of section 6.3, it can be concluded that the proposed approach performs better than the existing similar approaches. It was shown that creating a model per cluster rather than creating a model per host or creating a single model results in a higher detection accuracy. When applied to identifying malicious hosts, all clustering methods perform better, whereas for detection malicious flows only SBM clustering offers a consistently better detection performance.

When comparing the trade-off between detection rates and the processing burden, the proposed approach again is the best option. As the computational complexity of SVM equals $O(n^3)$ for n training samples[14], it is more efficient to have multiple SVM instances with some training samples than one SVM with all training samples. In other words, detection using one model is computationally demanding, while per-host clustering has a relatively low computational burden. Our approach is in between of these two in this respect. Therefore, the proposed approach decreases the computational burden compared to single-model detection, while improving the detection accuracy.

[14]See also section 5.3.1.

7 Conclusion

This research was motivated by the increasing complexity of targeted attacks. Misuse-based approaches are insufficiently capable of detecting new, unseen attacks and anomaly-based detection often suffers from high false positive rates. Moreover, many intrusion detection systems leave a valuable information source, internal network traffic, untouched. In order to improve current detection mechanism, our main research question was to investigate *how well internal network traffic anomaly detection based on host clustering performs in detecting network attacks such as* APT*s* - a new approach that has not been investigated before.

Firstly, we investigated the role of internal network traffic in Advanced Persistent Threats. It was shown that this class of threats has a significant impact on internal network traffic patterns. Although anomalous internal network behaviour can be used to identify malicious intents, this aspect is in current approaches rarely taken into account, leaving a valuable indicator of compromise unused. Another conclusion is that since the reviewed APT campaigns are unique in their methods, it is clear that signature-based solutions will not provide protection against this type of attack.

Secondly, host clustering as a means to improve anomaly detection approaches was studied. Hosts active on the internal network can be similar in various ways, mainly depending on which host characteristics are taken into account. Similarly, different host within an internal network can also show very different behaviour. We have shown that clustering only those hosts that show similar behaviour can indeed improve the detection algorithms. For the different algorithms available, there is a main distinction between methods that take value vectors into account and those that take connectivity into account. In the context of detecting APT-style campaigns, clustering methods that take connectivity into account perform best when it comes to anomaly detection.

Thirdly, it was explored how models can be derived from clusters of hosts, which describe normal behaviour of the hosts within a cluster. We argued that identifying anomalous behaviour on internal network level can be effectively done using captured host interaction statistics. Essential in this are the source and destination host, while additional statistics such as start time, duration and transfer size contribute to more reliably identifying unusual network patterns. Applying this to host clusters, the statistics of hosts in the same cluster show strong similarity.

Fourthly, this knowledge was used for a new approach that combines internal network patterns with host-clustering (through model-based detection). This is achieved by comparing the network traffic a host generates to the model of the cluster it belongs to. Unusual combinations of flow statistics are identified by rating each flow according to its level of the deviation from the norm.

Finally, the suggested approach was evaluated. The results show that the new approach has an improved detection rate both when identifying malicious flows as when identifying malicious hosts. Furthermore, the proposed approach reduces the burden of data processing through the applied clustering, compared to similar model-based approaches.

To sum up, the presented clustering approach can accurately model the behaviour of hosts which results in improved anomaly detection, providing a promising improvement within the field of advanced network intrusion detection. The results of this work may give rise to further research into the application of clustering within the context of anomaly detection.

8 Discussion and future work

This chapter discusses the limitations of this research as well as suggestions for future work.

8.1 Discussion

Four points of discussion regarding this research were identified and are presented in the following subsections.

8.1.1 Verification data

The main obstacle in this domain of research is the lack of data that is available for validation, as has also become apparent in this work. The available data sets suffer from various kinds of issues, narrowing the possibilities for a thorough evaluation of new intrusion detection methods in general. Additionally, the necessity for internal network traffic data has proven to even further reduce the number of relevant, available data sets, as a result of the limited research into this area thus far.

Yet, the data set used in this research satisfies all requirements for a proper evaluation, and the data supports the claims made regarding the effect of clustering on detection rates and the importance of monitoring internal network traffic. Nonetheless, a more extensive validation is recommended in order to further validate the findings of this work.

8.1.2 Future attacks

Questions might be raised concerning the extent to which the presented work is future-proof. It is arguable that, since it is only possible to look at attacks that took place in the past, the analysis of APTs only focuses historical data and therefore does not provide any basis for statements about future attacks. Especially for APTs, new attacks are likely to take different approaches - hence, one might question the sustainability of this work and its findings.

Indeed, in the context of APTs, it is practically impossible to make assertions about how effective an approach is towards APT campaigns in general. At most, we can make statements to what degree APTs from the past would have been detected by the proposed approach. In this work, we have tried to make this case stronger by analysing the attack characteristics of analysed APTs and determining what elements are distinctive for these attacks (see chapter 3). This approach enables us to reason about APT-style attack characteristics at a higher level rather than specific attacks.

However, this does not mean it is unnecessary to take new developments into account. APT detection and defence is a highly dynamic, innovative are of research that requires one to keep up with in order to be effective.

8.1.3 Number of clustering methods examined

As has also been discussed in chapter 4, it was only possible to evaluate a limited number of clustering methods for the purposes of this research.

For this reason, the four chosen clustering methods were deliberately chosen from different classes that require a different type of features. By this, it was possible to make statements about which type and form of clustering is most suitable for the purposes of network anomaly detection. In the end, the goal of this research was showing the added benefit of applying clustering, not to find the best clustering method for this purpose. This could be further investigated in future research (see section 8.2).

8.1.4 Reliability of input data

One of the assumptions that is made for the proposed solution is the availability of reliable data. For the evaluation of the concept, flow data was considered. However, it is not self-evident that this data can be reliably captured [31]. It is not ruled out that as a result of (slightly) inconsistent, unpredictable captures the results of our detection algorithm might get tainted as well. It is recommended that the effect of this phenomenon on the detection outcome is investigated further.

8.2 Future work

This research leads to some future work, which is discussed in the following subsections.

8.2.1 Alternative clustering methods

As we have seen in section 8.1, finding the best clustering method with the purpose of enhancing anomaly detection was outside the scope of this research. It might benefit existing anomaly detection solutions to further narrow down which specific clustering methods (potentially, under which parameters and assumptions) result in the best detection rates. This could include entirely different clustering methods than the four discussed in this work. Additionally, another interesting mode of clustering to investigate is fuzzy clustering, in which a single host can belong to multiple clusters at the same time with a certain probability.

8.2.2 Different data sources

While this research primarily concentrates on flow data, it is worthwhile to investigate the trade-off between the various data types. As discussed in section 5.1.2, there are clear differences between the level of information they provide. Still, since it remains unclear how the data source impacts the detection rates, it would be worthwhile to further explore whether choosing a different data source would improve the detection rates.

8.2.3 IPv6 compatibility

In the current approach, it is assumed the network data is carried over IPv4. For the sustainability of the presented approach, it should be investigated what implications the future deployment of IPv6 has with regards to this type of anomaly detection.

Although the flow data format will not require great change, the IPv6 protocol has some notable differences to its predecessor. As a result of protocol changes and the introduction of new functionality, assumptions that hold for IPv4 may no longer hold for IPv6. For instance, the assumption that an IP address can reliably be mapped to the same host is no longer trivial under IPv6, in which new alternatives to DHCP and the introduction of privacy extension may make tracking machines harder.

Additionally, a transition to IPv6 might result in changes in the behaviour of APTs. The steps taken on the internal network changes are likely to change: this might give opportunities for both sides, making it either easier or harder to detect intrusions. Although there has been research into this area [47], further research on the impact on internal network traffic specifically is recommended.

8.2.4 Broader application

The presented anomaly detection algorithm for internal network traffic is just an instance of the more general concept of applying clustering as a 'smart' intermediary step. In future work, the application of this concept in a broader context should be investigated.

For instance, it could be examined how clustering could enhance similar concepts (e.g. user behaviour analytics), how it can be applied to different data sources (e.g. external network traffic data) and domains (e.g. host-level monitoring), or even outside the intrusion detection domain (anomaly detection in the broadest sense).

A Proof-of-concept implementation

An important part of this research involved the development of a proof-of-concept of the anomaly detection system that is proposed in this thesis. This enabled us to make statements about the performance, efficiency and usability of the proposed methods. The proof-of-concept is based on the description of the method as in chapter 5 and is used for the evaluation in chapter 6.

This appendix briefly describes the structure of the proof-of-concept, the functionality of the modules and some of the inner workings. The proof-of-concept is written in the *Python* programming language and consists of four main modules, `generation`, `preprocessing`, `clustering` and `analysis`.

To get access to the full source code (including code annotations and usage examples) of this proof-of-concept, please contact the author.

A.1 Generating network traffic

The `generation` module can generate PCAP files according to a client-server model. It is possible to specify the number of servers and number of clients, upon which network data with those numbers of hosts is simulated. This module relies on `ns-3`[15], an free, open-source, discrete-event network simulator for Internet systems, targeted for research use.

Each client connects at least once to a number of servers, determined randomly based on a Pareto distribution. Subsequently, the number of flows, the number of bytes per flow, the duration per flow and the interval between flows are randomly generated according to statistical distributions too. The number of flows is decided following a Pareto distribution, whereas the others are generated according a Weibull distribution.

The module configures `ns-3` to generate traffic according to this infrastructure. Furthermore, the model aggregates the data produced by `ns-3`. The result is a single PCAP file, containing the generated network traffic. The contents of the packets are empty, i.e. null bytes.

This module is not essential to the proof-of-concept, but helps generating internal network traffic with the option to tune the parameters to alter the network configuration. This may be useful for the analysis carried out in the following modules.

A.2 Preprocessing

The second module focuses on preparing the data before it will be analysed by the next modules. This preprocessing entails filtering internal network traffic and transforming data into a

[15]Available at `https://www.nsnam.org/`.

common data format. Depending on the input data, it might also be necessary to convert the data to flow data.

Firstly, all traffic for which the source and destination IP address do match a given prefix, indicating that the communication is between internal hosts, are filtered. While for home networks this prefix usually is `192.168.0.0/16`, other networks might use different prefixes.

The `preprocessing` module includes scripts to transform full traffic captures, flow data and DNS data into a common data format as described in table A.1. Flow data can be extracted from XML files with a certain format by simply transforming the data. Full traffic captures and DNS data can be extracted from PCAP files, for which the tool `tshark`[16] is used. As this proof-of-concepts focuses on flow data, PCAP files are aggregated into flows. The definition of flows is determined by `tshark`, in which for TCP traffic a flow is defined as a connection (from establishment to termination) and for UDP traffic as a single datagram.

Table A.1: Common data format used for this proof-of-concept. Each record represents a single flow.

Property	Type	Description
`Address_A`	String	IP Address of the initiator (source) host.
`Port_A`	String	Port number of the initiator (source) host.
`Address_B`	String	IP Address of the destination host.
`Port_B`	String	Port number of the destination host.
`Packets`	Integer	The total number of packets sent in this flow.
`Bytes`	Integer	The total number of bytes sent in this flow.
`Packets_A→B`	Integer	The number of packets sent from the source to the destination host.
`Bytes_A→B`	Integer	The number of bytes sent from the source to the destination host.
`Packets_A←B`	Integer	The number of packets sent from the destination to the source host.
`Bytes_A←B`	Integer	The number of bytes sent from the destination to the source host.
`Rel_Start`	Float	The relative start time of this flow. The value for the first record is 0.
`Duration`	Float	The duration of the flow.
`bps_A→B`	Float	The transfer rate in bytes per second of the source host.
`bps_A←B`	Float	The transfer rate in bytes per second of the destination host.
`Tag`	String	(Optional) Indicator used for labelling. should be either `Normal` or `Attack`. Other values will be ignored.

A.3 Clustering

The `clustering` module is used for clustering networks. Table A.2 provides an overview of the implemented clustering methods for this module, their dependencies and non-default configuration options. For k-means and mean shift clustering, the following feature set was used: [COUNT($Incoming\_Connections$), COUNT($Outgoing\_Connections$), AVG($Duration_{In}$), AVG($Duration_{Out}$), AVG($Bytes_{In}$), AVG($Bytes_{Out}$), AVG($Packets_{In}$), AVG($Packets_{Out}$)].

[16]See https://www.wireshark.org/docs/man-pages/tshark.html.

Given a data file (in the format as described in section A.2) of a network and a clustering method, this module establishes the clustering of the network. This is presented as a mapping between each unique IP address to the cluster identifier ($\in [0, k\rangle$ for k clusters) the host belongs to. The module allows setting certain cluster-specific parameters that may impact the clustering outcome.

Additionally, this module can generate visualisation of the graphs in various formats, it can generate time-lapses for a given network over time, it can compute statistics of the data files based on the clustering found and it can compute ARI scores for two given clusterings.

Table A.2: Implemented clustering methods and the packages they depend on.

Type	**Implementation/package**	**Non-default configuration**
k-means clustering	`sklearn` [53]	Feature set
Mean shift clustering	`sklearn` [53]	Feature set
Louvain clustering	`community`[17]	Weighted
SBM Clustering	`graph_tool` [56]	Non-degree corrected, weighted

A.4 Analysis

The fourth and final module `analysis` applies anomaly detection to the data files after clustering. This module consists of two major parts: one in which models are created, another in which new data files can be compared using SVM to the created models.

For creating new models, test data (in form of a data file in the common data format) and a clustering method have to be supplied. The test data is first clustered, after which for each cluster the relevant flow data is added to an SVM. This proof-of-concept is based on the `sklearn` [53] implementation of SVMs. The following feature set was used: [`Bytes_A→B`, `Bytes_A←B`, `Packets_A→B`, `Packets_A←B`, `Duration`, `Rel_Start`, `Cluster_B`], the last feature being the cluster to which the destination host belongs. This latter is the only categorical feature. The clustering together with the corresponding SVM instances are saved to a single file.

For detection anomalies, training data (in form of a data file in the common data format) and a model file (as described in the previous paragraph) have to be supplied. After loading the test data and the models, each data flow is provided to the SVM corresponding to the cluster the relevant host belongs to, resulting in an anomaly score. At the end, the anomaly scores are summed up for each host and represented as a report, which can be used for further analysis. If labelling is provided in the test data, this module can also generate ROC curves.

[17] Available at `http://perso.crans.org/aynaud/communities/`.

Bibliography

[1] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, 9(Sep):1981–2014, 2008.

[2] B. Bencsáth, G. Pék, L. Buttyán, and M. Félegyházi. Duqu: Analysis, detection, and lessons learned. In *ACM European Workshop on System Security (EuroSec)*, volume 2012, 2012.

[3] D. K. Bhattacharyya and J. K. Kalita. *Network Anomaly Detection: A Machine Learning Perspective*. Chapman & Hall/CRC, 2013. ISBN 1466582081, 9781466582088.

[4] V. Blondel. The Louvain method for community detection in large networks, 2011. URL `https://perso.uclouvain.be/vincent.blondel/research/louvain.html`. Accessed on 14/04/2016.

[5] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10): P10008, 2008. doi:10.1088/1742-5468/2008/10/P10008.

[6] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[7] R. Brewer. Advanced Persistent Threats: Minimising the Damage. *Network Security*, 2014 (4):5 – 9, 2014. ISSN 1353-4858. doi:10.1016/S1353-4858(14)70040-6.

[8] C. Brown, A. Cowperthwaite, A. Hijazi, and A. Somayaji. Analysis of the 1999 darpa/lincoln laboratory ids evaluation data with netadhict. In *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*, pages 1–7. IEEE, 2009.

[9] M. D. Byrne. How many times should a stochastic model be run - An approach based on confidence intervals. In *Proceedings of the 12th International conference on cognitive modeling*, Ottawa, 2013.

[10] CAIDA. The CAIDA UCSD Passive Data Sets, 2016. URL `https://www.caida.org/data/passive/`. Accessed on 29/4/2016.

[11] S. Chebrolu, A. Abraham, and J. P. Thomas. Hybrid feature selection for modeling intrusion detection systems. In *Neural Information Processing*, pages 1020–1025. Springer, 2004.

[12] S. Chen, M. Peng, H. Xiong, and X. Yu. SVM Intrusion Detection Model Based on Compressed Sampling. *Journal of Electrical and Computer Engineering*, 2016, 2016.

[13] M. Claesen, F. De Smet, J. A. Suykens, and B. De Moor. Fast prediction with SVM models containing RBF kernels. *arXiv:1403.0736*, 2014.

[14] B. Claise, J. Quittek, J. Meyer, S. Bryant, and P. Aitken. Information Model for IP Flow Information Export, Oct. 2015. doi:10.17487/rfc5102.

[15] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002. ISSN 0162-8828. doi:10.1109/34.1000236.

[16] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 142–149. IEEE, 2000. doi:10.1109/CVPR.2000.854761.

[17] R. Danyliw, J. Meijer, and Y. Demchenko. The Incident Object Description Exchange Format. RFC 5070 (Proposed Standard), December 2007. doi:10.17487/rfc5070.

[18] J. J. Davis and A. J. Clark. Data preprocessing for anomaly based network intrusion detection: A review. *Computers & Security*, 30(6-7):353 – 375, 2011. ISSN 0167-4048. doi:10.1016/j.cose.2011.05.008.

[19] DEF CON. DEF CON downloads per edition, 2016. URL `https://www.defcon.org/html/links/dc-torrent.html`. Accessed on 29/3/2016.

[20] Dell SecureWorks. Advanced persistent threat analysis, 2013. URL `http://www.secureworks.com/cyber-threat-intelligence/advanced-persistent-threat/understand-the-threat/`. Accessed on 21/01/2016.

[21] D. E. Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering*, 13(2):222–232, 1987. doi:10.1109/TSE.1987.232894.

[22] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A geometric framework for unsupervised anomaly detection. In *Applications of data mining in computer security*, pages 77–101. Springer, 2002.

[23] N. Falliere, L. O. Murchu, and E. Chien. W32.Stuxnet Dossier, 2011. URL `http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf`. Accessed on 9/11/2015.

[24] J. P. Farwell and R. Rohozinski. Stuxnet and the Future of Cyber War. *Survival*, 53(1):23–40, 2011. doi:10.1080/00396338.2011.555586.

[25] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010. ISSN 0370-1573. doi:10.1016/j.physrep.2009.11.002.

[26] P. Garcìa-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 28(1-2):18 – 28, 2009. ISSN 0167-4048. doi:10.1016/j.cose.2008.08.003.

[27] A. A. Ghorbani, W. Lu, and M. Tavallaee. *Network intrusion detection and prevention: concepts and techniques*, volume 47. Springer Science & Business Media, 2009.

[28] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002. doi:10.1073/pnas.122653799.

[29] A. Gostev. The Flame: Questions and Answers, 2012. URL `https://securelist.com/blog/incidents/34344/the-flame-questions-and-answers-51/`. Accessed on 19/02/2016.

[30] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011. ISBN 0123814790, 9780123814791.

[31] R. J. Hofstede, I. Drago, A. Sperotto, R. Sadre, and A. Pras. Measurement artifacts in netflow data. In *14th International Conference on Passive and Active Measurement, PAM 2013, Hong Kong, China*, volume 7799 of *Lecture Notes in Computer Science*, pages 1–10, Berlin, March 2013. Springer Verlag. ISBN 978-3-642-36515-7. doi:10.1007/978-3-642-36516-4_1.

[32] R. J. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras. Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX. *IEEE Communications Surveys Tutorials*, 16(4):2037–2064, 2014. ISSN 1553-877X. doi:10.1109/COMST.2014.2321898.

[33] P. W. Holland, K. B. Laskey, and S. Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5(2):109 – 137, 1983. ISSN 0378-8733. doi:10.1016/0378-8733(83)90021-7.

[34] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985. ISSN 1432-1343. doi:10.1007/BF01908075.

[35] E. M. Hutchins, M. J. Cloppert, and R. M. Amin. Intelligence-driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains. *Leading Issues in Information Warfare & Security Research*, 1:80, 2011.

[36] A. K. Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8): 651–666, 2010. doi:10.1016/j.patrec.2009.09.011.

[37] A. K. Jain, A. Topchy, M. H. C. Law, and J. M. Buhmann. Landscape of Clustering Algorithms. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 1 - Volume 01*, ICPR '04, pages 260–263, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2128-2. doi:10.1109/ICPR.2004.525.

[38] Kaspersky. Carbanak APT: The Great Bank Robbery, 2015. URL `https://securelist.com/files/2015/02/Carbanak_APT_eng.pdf`. Accessed on 18/2/2016.

[39] S. S. Keerthi and C.-J. Lin. Asymptotic behaviors of support vector machines with gaussian kernel. *Neural computation*, 15(7):1667–1689, 2003. doi:10.1162/089976603321891855.

[40] A. Kind, M. P. Stoecklin, and X. Dimitropoulos. Histogram-based traffic anomaly detection. *Network and Service Management, IEEE Transactions on*, 6(2):110–121, 2009. doi:10.1109/TNSM.2009.090604.

[41] R. Langner. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security and Privacy*, 9(3): 49–51, 2011. doi:10.1109/MSP.2011.67.

[42] A. Lazarevic, V. Kumar, and J. Srivastava. Intrusion detection: A survey. In V. Kumar, J. Srivastava, and A. Lazarevic, editors, *Managing Cyber Threats: Issues, Approaches, and Challenges*, pages 19–78. Springer US, Boston, MA, 2005. ISBN 978-0-387-24230-9. doi:10.1007/0-387-24230-9_2.

[43] W. Lee and S. J. Stolfo. A framework for constructing features and models for intrusion detection systems. *ACM transactions on Information and system security (TiSSEC)*, 3(4): 227–261, 2000.

[44] Y. Li, J.-L. Wang, Z.-H. Tian, T.-B. Lu, and C. Young. Building lightweight intrusion detection system using wrapper-based feature selection mechanisms. *Computers & Security*, 28(6): 466 – 475, 2009. ISSN 0167-4048. doi:10.1016/j.cose.2009.01.001.

[45] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1(14), pages 281–297. Oakland, CA, USA., 1967.

[46] McAfee. Net Losses: Estimating the Global Cost of Cybercrime. *McAfee, Centre for Strategic & International Studies*, 2014.

[47] B. J. Menšık. *IPv6 Impact on Network Intrusion Detection.* PhD thesis, Czech Technical University, Prague, 2012.

[48] G. Münz, S. Li, and G. Carle. Traffic anomaly detection using k-means clustering. In *GI/ITG Workshop MMBnet*, 2007.

[49] K. Nowicki and T. A. B. Snijders. Estimation and Prediction for Stochastic Blockstructures. *Journal of the American Statistical Association*, 96(455):1077–1087, 2001. doi:10.1198/016214501753208735.

[50] I.-V. Onut and A. A. Ghorbani. *Information Security: 10th International Conference, ISC 2007, Valparaíso, Chile, October 9-12, 2007. Proceedings*, chapter Features vs. Attacks: A Comprehensive Feature Selection Model for Network Based Intrusion Detection Systems, pages 19–36. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. ISBN 978-3-540-75496-1. doi:10.1007/978-3-540-75496-1_2.

[51] R. Pang, M. Allman, M. Bennett, J. Lee, V. Paxson, and B. Tierney. A first look at modern enterprise traffic. In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, pages 2–2. USENIX Association, 2005.

[52] A. Patel, M. Taghavi, K. Bakhtiyari, and J. C. Júnior. An intrusion detection and prevention system in cloud computing: A systematic review. *Journal of Network and Computer Applications*, 36(1):25–41, 2013. doi:10.1016/j.jnca.2012.08.007.

[53] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[54] T. P. Peixoto. Parsimonious Module Inference in Large Networks. *Phys. Rev. Lett.*, 110: 148701, Apr 2013. doi:10.1103/PhysRevLett.110.148701.

[55] T. P. Peixoto. Efficient Monte Carlo and greedy heuristic for the inference of stochastic block models. *Phys. Rev. E*, 89:012804, Jan 2014. doi:10.1103/PhysRevE.89.012804.

[56] T. P. Peixoto. The graph-tool Python library. *Figshare*, 2014. doi:10.6084/m9.figshare.1164194. URL `http://figshare.com/articles/graph_tool/1164194`.

[57] D. Pelleg, A. W. Moore, et al. X-means: Extending K-means with Efficient Estimation of the Number of Clusters. In *ICML*, pages 727–734, 2000. ISBN 1-55860-707-2.

[58] M. A. Porter, J.-P. Onnela, and P. J. Mucha. Communities in networks. *Notices of the AMS*, 56(9):1082–1097, 2009.

[59] W. M. Rand. Objective Criteria for the Evaluation of Clustering Methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971. doi:10.1080/01621459.1971.10482356.

[60] Research and Markets. Advanced persistent threat protection market - global forecast to 2020, dec 2015.

[61] F. Sabahi and A. Movaghar. Intrusion Detection: A Survey. In *Systems and Networks Communications, 2008. ICSNC '08. 3rd International Conference on*, pages 23–26, Oct 2008. doi:10.1109/ICSNC.2008.44.

[62] K. A. Scarfone and P. M. Mell. SP 800-94. Guide to Intrusion Detection and Prevention Systems (IDPS). Technical report, National Institute of Standards & Technology, Gaithersburg, MD, United States, 2007.

[63] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001. doi:10.1162/089976601750264965.

[64] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*, 31(3):357 – 374, 2012. ISSN 0167-4048. doi:10.1016/j.cose.2011.12.012.

[65] Symantec. W32.Flamer: Microsoft Windows Update Man-in-the-Middle, 2012. URL `http://www.symantec.com/connect/blogs/w32flamer-microsoft-windows-update-man-middle`. Accessed on 19/02/2016.

[66] Symantec. Threat Report 2015, 2016. URL `https://www4.symantec.com/mktginfo/whitepaper/ISTR/21347932_GA-internet-security-threat-report-volume-20-2015-social_v2.pdf`. Accessed on 07/03/2016.

[67] M. Tavallaee, E. Bagheri, W. Lu, and A.-A. Ghorbani. A detailed analysis of the KDD CUP 99 data set. In *Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications 2009*, 2009. doi:10.1109/CISDA.2009.5356528.

[68] C. Torrano Giménez et al. *Study of stochastic and machine learning techniques for anomaly-based Web atack detection.* PhD thesis, University Carlos III of Madrid, 2015.

[69] N. Virvilis and D. Gritzalis. The Big Four - What We Did Wrong in Advanced Persistent Threat Detection? In *Availability, Reliability and Security (ARES), 2013 Eighth International Conference on*, pages 248–254, Sept 2013. doi:10.1109/ARES.2013.32.

[70] R. Xu, D. Wunsch, et al. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678, 2005. doi:10.1109/TNN.2005.845141.

[71] Y. Zhao, E. Levina, and J. Zhu. Consistency of community detection in networks under degree-corrected stochastic block models. *Ann. Statist.*, 40(4):2266–2292, 08 2012. doi:10.1214/12-AOS1036.

[72] H. Zimmermann. OSI Reference Model–The ISO Model of Architecture for Open Systems Interconnection. *Communications, IEEE Transactions on*, 28(4):425–432, Apr 1980. ISSN 0090-6778. doi:10.1109/TCOM.1980.1094702.